Universidad de Sevilla
Escuela Técnica Superior de Ingeniería
Departamento de Ingeniería Mecánica y de Fabricación

Proyecto fin de grado

Instrumentación de un vehículo ferroviario para la auscultación dinámica con sistemas de visión artificial

Autor: Juan Manuel Amador Olivares
Tutor: José Luis Escalona Franco

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Dedicado a
mi familia

II

Agradecimientos

Quiero agredecer en primer lugar a mi novia por su apoyo y su interés en el desarrollo del proyecto.

A mi tutor por haberme guiado y ayudado en todo lo que ha podido.

También quiero agredecerle a mis compañeros de trabajo todos sus consejos y la ayuda que siempre están dispuestos a dar.

A mi familia, porque su apoyo incondicional es lo más valioso que tengo y lo que me da fuerzas para luchar por alcanzar cualquier meta.

Resumen

En este trabajo final de grado se presenta un sistema de adquisición de datos embarcado en un vehículo ferroviario para proyectos experimentales de auscultación de vías.

El sistema de adquisición integra sensores inerciales para la auscultación dinámica, y cámaras de vídeo que permiten el uso de técnicas de visión artifical en la tarea de la auscultación de vías.

Se describirá la problemática de la auscultación, se desarrollará el problema cinemático asociado a la auscultación de vías con cámaras de vídeo y se describirán los sistemas de adquisición, tanto los sensores elegidos, como su diseño electrónico y programación.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El ferrocarril tiene numerosas ventajas frente a otros medios de transporte, a saber: capacidad, eficiencia, velocidad, seguridad, menor ocupación del suelo y menor contaminación.

La sencilla electrificación de este medio de transporte tenderá a aumentar su uso a medida que disminuya la disponibilidad de combustibles fósiles en las siguientes décadas.

1.1. Auscultación de vías

Esta sección está basada en una memoria del proyecto *Desarrollo de nuevas tecnologías de auscultación ferroviaria basadas en la simulación en tiempo real* [6].

El ferrocarril requiere de una insfraestructura en buenas condiciones cuyo mantenimiento es muy costoso reduciendo la eficiencia tanto energética como económica de este medio de transporte.

Un elemento con alto costo de mantenimiento de dicha infraestructura son las vías, que pueden extenderse hasta cientos de miles de kilómetros. Su auscultación (evaluar en las condiciones en las que se encuentra) es imprescindible para mantener la seguridad y el confort de este medio de transporte.

Durante la auscultación se mide la geometría de los carriles. Existen dos tipos de medidas en la auscultación: geométricas y dinámicas. Las geométricas miden las desviaciones con respecto a una geometría de referencia de los carriles. Las dinámicas miden la respuesta dinámica del vehículo ante dichas irregularidades.

La figura 1.1 muestra el sistema de referencia respecto al que se miden las irregularidades de la vía. Este sistema de referencia se sitúa en cada uno de los puntos de la línea media de la vía de referencia. Cada carril tendrá su

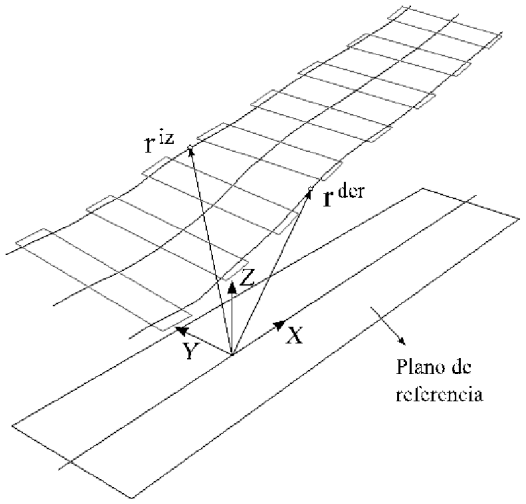

Figura 1.1: Medida de irregularidad de los carriles [6]

propio vector de irregularidad con coordenadas x, y, z.

Los defectos de los carriles se clasifican por las desviaciones de la geometría de estos, de acuerdo a los siguientes parámetros:

1. **Alineamiento**. De define como $(y^{izq} + y^{der})/2$.

2. **Nivelación**. De define como $(z^{izq} - z^{der})/2$.

3. **Ancho de vía**. De define como $(y^{izq} - y^{der})/2$.

4. **Pefil vertical**. De define como $(z^{izq} + z^{der})/2$.

5. **Peralte**. En los tramos curvos de la vía se eleva la parte exterior respecto a la interior, creándose un ángulo respecto a la horizontal. Por tanto, el peralte peude ser interpretado como un "defecto"de nivelación intencionado en los tramos curvos.

6. **Alabeo**. Ritmo de cambio de la nivelación de los carriles.

7. **Desgaste de perfil**. Diferencia entre el perfil del carril real y el ideal.

Las irregularidades pueden ser puntuales (como el caso de una frenada brusca). Sin embargo, suelen tener asociada una longitud de onda carácterística. Conocer la magnitud de las irregularidades así como la frecuencia (longitud de onda) es de vital importancia para el diseño un sistemas de adquisición de datos cuya función sea medir estas irregularidades.

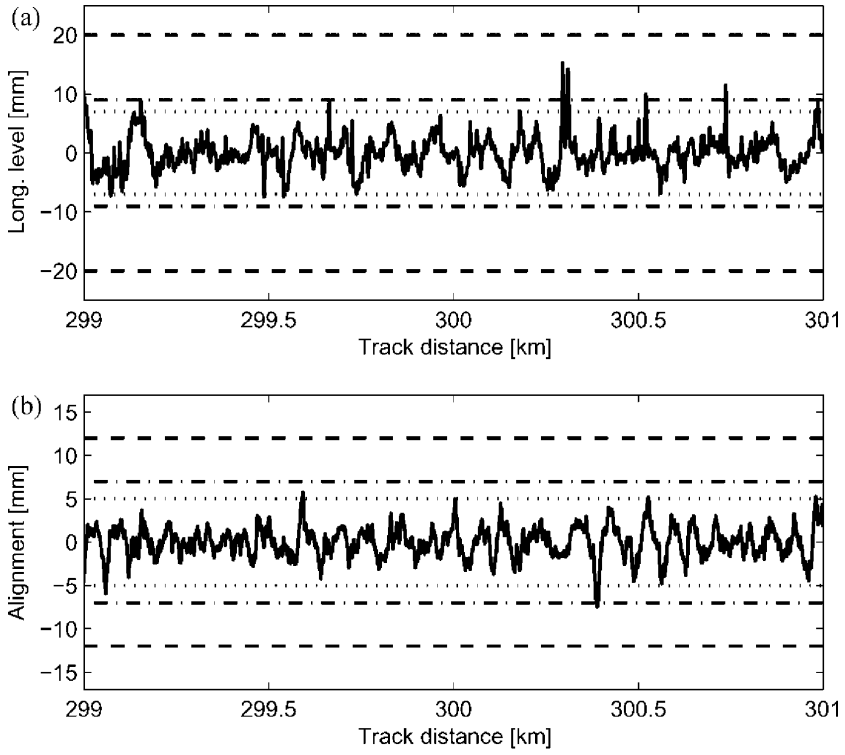

Figura 1.2: Medidas de los defectos en función de la distancia [7]

En la figura 1.2 puede observarse la magnitud de la nivelación y el alineamiento en función de la distancia recorrida por el vehículo. Es posible conocer la longitud de onda característica mediante el espectro de potencia (*Power Espectral Density*, PSD) calculado con la FFT (*Fast Fourier Transform*), este espectro se muestra en la figura 1.3.

Normalmente los defectos de alineamiento y nivelación suelen tener una PSD aproximadamente lineal, siendo mayor la amplitud del defecto para mayores longitudes de onda. Como puede observarse en las gráficas de la figura 1.3 las longitudes de onda de interés de las irregularidades es de 1 m debido a que por debajo de esta longitud de onda las irregularidades son muy pequeñas.

En este TFG se va a describir el sistema de adquisición de datos del sistema embarcado que realice la auscultación mediante el uso de sensores inerciales y técnicas de visión artificial, y se va a introducir el problema matemático de la auscultación dinámica de vías con el uso de visión artificial. Este sistema se ha fabricado, realizándose con él varios ensayos con vehículos ferroviarios.

El sistema de adquisición de datos será el encargado de sincronizar los

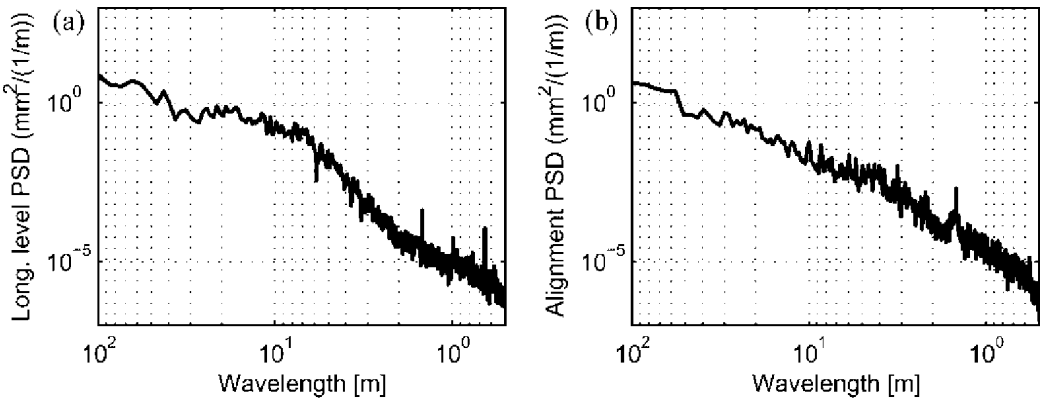

Figura 1.3: Espectro de potencia de los defectos [7]

distintos sensores y controlar el flujo de datos que serán procesados por el sistema embarcado.

1.2. Respuesta dinámica del vehículo

Para conocer las irregularidades de la vía el sistema embarcado procesa los datos de la dinámica del vehículo. Esta información puede ser utilizada con otros fines de gran importancia, como la prevención de accidentes o la medición del confort.

Capítulo 2

Formulación del problema

Este capítulo está basado en el trabajo de José Luís Escalona Franco, *Cinemática ferroviaria para su aplicación a sistemas de visión artificial* [5].

El problema que se plantea para la obtención de la dinámica vehicular y las irregularidades de la vía es fundamentalmente cinemático.

En primer lugar, se van a describir las ecuaciones utilizadas de visión artificial. En segundo lugar, se formularán las ecuaciones que describen la cinemática vehicular. Una vez conocidas las distintas ecuaciones que rigen la cinemática, se formulará el sistema de ecuaciones completo del problema. Por último, se describirá el algortimo que obtiene la información de la posición y orientación de los carriles grabados por las cámaras de vídeo a partir de las imágenes grabadas por estas.

2.1. Cinemática de la visión artificial con proyección láser lineal

El uso de cámaras de vídeo es indispensable para una reconstrucción y caracterización precisa de las irregularidades de las vías y el desgaste del perfil de los carriles.

El modelo de cámara utilizado es el de cámara estenopeica o cámara oscura tal y como puede observarse en la figura

La posición $\mathbf{v}_P^{cam}$ es la posición real de un punto del carril respecto al sistema de referencia de la cámara $< x^{cam}, y^{cam}, z^{cam} >$. La posición $\mathbf{v}_{P'}^{im}$ es la posición de la proyección del punto P en el plano sensor respecto al sistema de referencia de la imagen $< x^{im}, y^{im}, f >$. Estas dos posiciones pueden relacionarse mediante la ecuación 2.1, deducida del modelo de cámara oscura de la cámara y los triángulos semejantes que forman las componentes de los vectores $\mathbf{v}_P^{cam}$ y $\mathbf{v}_{P'}^{im}$.

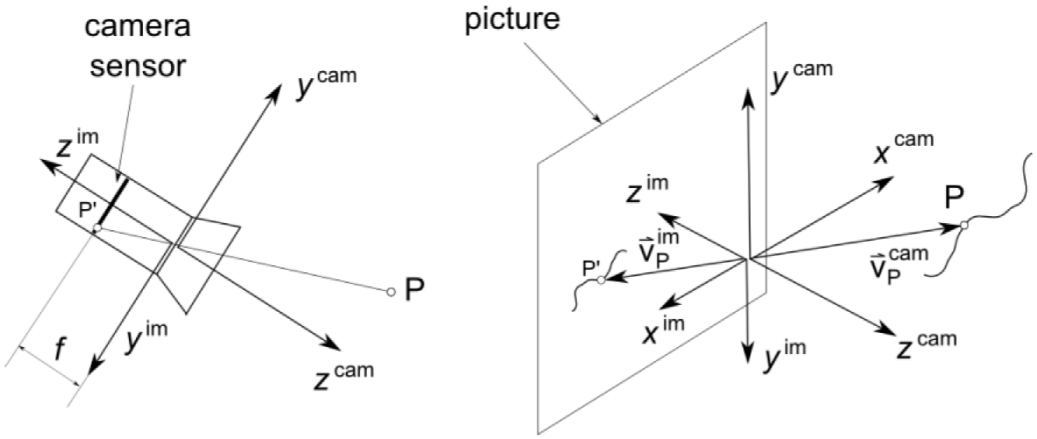

Figura 2.1: Cinemática de la visión artificial [5]

$$\frac{(\mathbf{v}_P^{cam})_z}{f} = \frac{(\mathbf{v}_P^{cam})_x}{(\mathbf{v}_{P'}^{im})_x} = \frac{(\mathbf{v}_P^{cam})_y}{(\mathbf{v}_{P'}^{im})_y} \tag{2.1}$$

donde $\mathbf{v}_P^{cam} = [(\mathbf{v}_P^{cam})_x, (\mathbf{v}_P^{cam})_y, (\mathbf{v}_P^{cam})_z]^T$ y $\mathbf{v}_{P'}^{im} = [(\mathbf{v}_{P'}^{im})_x, (\mathbf{v}_{P'}^{im})_y, (\mathbf{v}_{P'}^{im})_z]^T$ son las componentes de los puntos P, en la realidad, y P', en la imagen, respectivamente. La ecuación 2.1 puede reescribirse de la forma:

$$\frac{f}{(\mathbf{v}_P^{cam})_z}\mathbf{v}_P^{cam} = \mathbf{v}_{P'}^{im} \tag{2.2}$$

Las componentes x e y de $\mathbf{v}_{P'}^{im}$ son datos que se pueden extraer de la imagen grabada de la siguiente forma:

$$\begin{aligned}(\mathbf{v}_{P'}^{im})_x &= a(n_x - n_{x_0}) \\ (\mathbf{v}_{P'}^{im})_y &= b(n_y - n_{y_0})\end{aligned} \tag{2.3}$$

donde a y b son la anchura y altura de los píxeles de la cámara respectivamente. n_x y n_y son las posiciones x e y del píxel a procesar. Las constantes n_{x_0} y n_{y_0} indican la posición del centro de la imagen (donde se encuentra el origen del sistema de coordenadas de la imagen).

La ecuación 2.2 contiene dos ecuaciones independientes para el cálculo de las tres componentes de $\mathbf{v}_P^{cam}$ (sistema indeterminado). De la imagen grabada por una cámara (2D) no se puede extraer la posición de los puntos que aparecen en el espacio (3D). Una técnica común para atajar este problema es utilizar ilumminación estructurada. Este tipo de iluminación se sirve de la proyección de puntos, franjas o rejillas sobre la superficie de trabajo. En

función de cómo se deforme este patrón de luz sobre la superficie se puede detectar las singularidades de la pieza objeto de análisis.

En este proyecto hemos conseguido la iluminación estructurada utilizando un láser de proyección lineal que genera un haz plano de luz además de la cámara. Gracas al uso del láser se tiene la información de que todos los puntos iluminados por el láser están contenidos en un plano, es decir, que los vectores posición de dichos puntos cumplen la siguiente condición:

$$a(\mathbf{v}_Q^{cam})_x + b(\mathbf{v}_Q^{cam})_y + c(\mathbf{v}_Q^{cam})_z - d = 0 \Rightarrow [a,b,c] \begin{bmatrix} \mathbf{v}_Q^{cam})_x \\ \mathbf{v}_Q^{cam})_y \\ \mathbf{v}_Q^{cam})_z \end{bmatrix} - d = 0$$
$$\Rightarrow \triangle^T \mathbf{v}_Q^{cam} - d = 0 \quad (2.4)$$

donde a, b, c, y d son los parámetros de la ecuación general del plano del láser y $\mathbf{v}_Q^{cam}$ es el vector posición de los puntos Q iluminados por el láser en el sistema de referencia de la cámara.

Cuando la cámara y el láser son estacionarios, las dos ecuaciones independientes que se extraen de la ecuación 2.2 junto con la ecuación 2.4 dan lugar a un sistema determinado de ecuaciones (tres ecuaciones y tres incógnitas) del que se puede extraer la posición (3D) $\mathbf{v}_P^{cam}$ de los puntos grabados en la imagen e iluminados por el láser. El problema que se quiere resolver en el presente proyecto es mucho más complejo que este porque:

1. Ni la cámara ni el láser son estacionarios, sino que están fijos a un vehículo en movimiento, por lo que la orientación del objeto grabado respecto a la cámara y el láser puede cambiar con el tiempo.

2. El objeto cuyas coordenadas se pretende identificar no esta fijo en el espacio ni con respecto al vehículo y además tiene geometría cambiante. Este objeto es la sección transversal del carril ferroviario.

3. A partir de la imagen grabada se pretende obtener, además de la gemetría de la sección transversal del carril, la medida de los movimientos del vehículo con respecto a la vía. Además, debe tenerse en cuenta que el deplazamiento de los carriles respecto al vehículo grabados por la cámara pueden deberse a la propia dinámica del vehículo o a las irregularidades de los carriles. Ambos problemas están acoplados.

Los aspectos matemáticos relacionados con estos movimientos y geometrías se exponen en la figura 2.2. En ella se muestran un único láser y una cámara. Sin embargo, el vehículo tendrá dos sistemas de visión como este, de manera que habrá uno para cada carril de la vía.

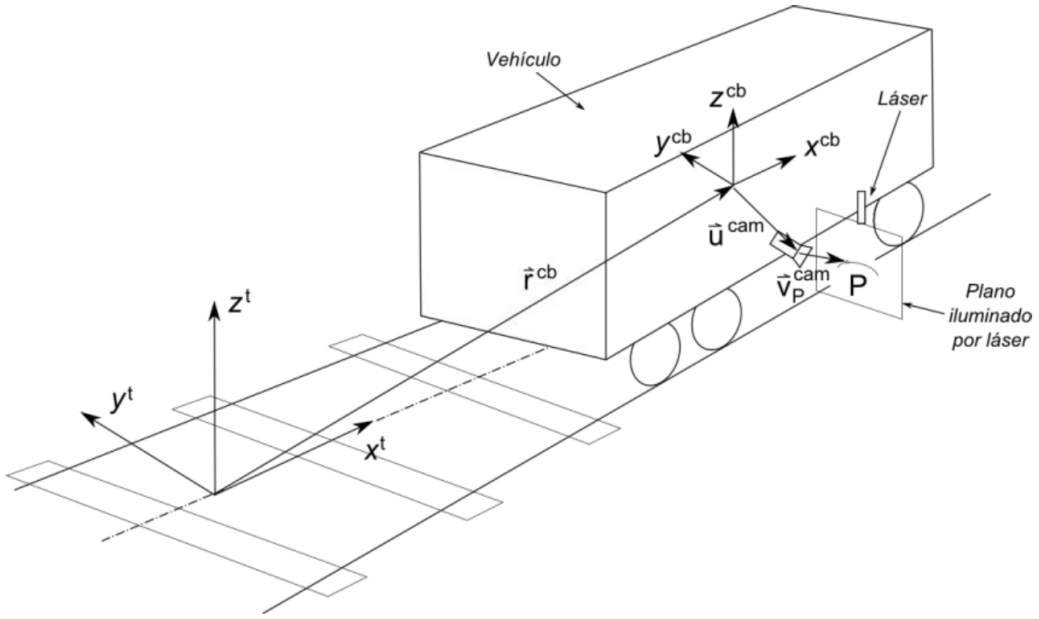

Figura 2.2: Cinemática del vehículo instrumentado [5]

2.2. Cinemática vehicular

La cinemática vehicular, las irregularidades de la vía y las imágenes grabadas de los carriles están relacionadas entre ellas. En esta sección se desarrollarán las ecuaciones que nos permitirán (junto a la cinemática de la visión artificial) formular el sistema de ecuaciones que se van a resolver.

2.2.1. Cinemática de los puntos filmados desde un vehículo en movimiento

La figura 2.2 muestra esquemáticamente la cámara y el láser instalados en el vehículo. Para determinar la dinámica del vehículo en cualquier instante (uno de los objetivos del sistema embarcado), es necesario conocer la posición y orientación (tres coordenadas de posición: $\mathbf{r}_x^{cb}$, $\mathbf{r}_y^{cb}$, $\mathbf{r}_z^{cb}$, y tres coordenadas de orientación: φ^{cb}, θ^{cb}, ψ^{cb}, *roll*, *pitch* y *yaw* respectivamente) del sistema de referencia del vehículo $< x^{cb}, y^{cb}, z^{cb}$ (cb $\equiv$ *car body*) con respecto al sistema de referencia de la vía $< x^t, y^t, z^t >$ (t $\equiv$ *track*). Se supone que el sistema de referencia de la vía se mueve con la misma velocidad que el vehículo a lo largo de la línea central de la vía con geometría ideal (sin verse modificada por las irregularidades). Las variaciones de las coordenadas $\mathbf{q}^{cb} = [\mathbf{r}_x^{cb}, \mathbf{r}_y^{cb}, \mathbf{r}_z^{cb}, \varphi^{cb}, \theta^{cb}, \psi^{cb}]$ son debidas por tanto a las irregularidades de la vía y a la

dinámica del vehículo (en respuesta, entre otras cosas, a la irregularidad). Se puede asumir sin perdida de generalidad que el sistema de referencia de la vía acompaña longitudinalmente al sistema de referencia del vehículo, es decir, que $x^{cb} = 0$.

En el planteamiento que sigue se supone que son conocidas la posición respecto al *car body* ($\overline{\mathbf{u}}^{cam}$) y orientación de la cámara con respecto al sistema de referencia del vehículo, así como los parámetros de la ecuación general del plano proyectado por el láser con respecto al sistema de referencia del vehículo. La posición de los puntos P iluminados por el láser y grabados por la cámara en el sistema de referencia de la vía se puede obtener como:

$$\mathbf{r}_P = \mathbf{r}^{cb} + \mathbf{A}^{t,cb}\left[\overline{\mathbf{u}}^{cam} + \mathbf{A}^{cb,cam}\mathbf{v}_P^{cam}\right] \tag{2.5}$$

donde $A^{t,cb}$ y $A^{cb,cam}$ son las matrices de rotación del vehículo con respecto a la vía (variable) y de la cámara con respecto al vehículo (constante), respectivamente. Debido a que los ángulos φ^{cb}, θ^{cb}, ψ^{cb} son muy pequeños, la matriz de orientación $A^{t,cb}$ se puede aproximar como:

$$A = \begin{pmatrix} 1 & -\psi^{cb} & \theta^{cb} \\ \psi^{cb} & 1 & -\varphi^{cb} \\ -\theta^{cb} & \varphi^{cb} & 1 \end{pmatrix} \tag{2.6}$$

En el apéndice A sección A.1 se han detallado las hipótesis y el desarrollo de obtención de esta matriz de rotación aproximada.

2.2.2. Cinemática de la vía irregular

Las figuras 5 y 6 muestran las relaciones cinemáticas de la vía irregular. La posición de cada uno de los carriles es función del desplazamiento lateral y vertical del carril debido a la irregularidad con respecto a la posición en la curva ideal. Estas son las dos componentes de los vectores $\mathbf{r}_{left}^{irr}$ y $\mathbf{r}_{right}^{irr}$ que se muestran en la figura 2.3.

La posición de un punto de la superficie de la cabeza del carril (por ejemplo, el carril izquierdo) con respecto al sistema de referencia de la vía se puede obtener de la siguiente forma:

$$\mathbf{r}_P = \mathbf{r}_0^{rp} + \mathbf{r}_{left}^{irr} + \mathbf{A}^{t,rp}\overline{\mathbf{u}}_P^{rp} \tag{2.7}$$

donde $\mathbf{r}_0^{rp} = [0, L_r 0]$ es la posición del sistema de referencia del carril izquierdo con respecto al sistema de referencia de la vía en ausencia de irregularidad, $\mathbf{r}_{left}^{irr} = [0, (\mathbf{r}_{left}^{irr})_y, (\mathbf{r}_{left}^{irr})_z]^T$ es el vector de irregularidad del carril izquierdo, $\mathbf{A}^{t,rp}$ es la matriz de rotación del sistema de referencia de la sección transversal del carril irregular $< x^{rp}, y^{rp}, z^{rp} >$ (rp $\equiv$ rail profile) con respecto

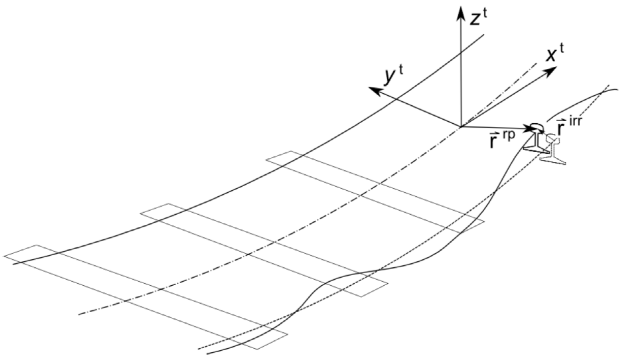

Figura 2.3: Cinemática de la vía irregular [5]

al sistema de referencia de la vía, y $\overline{\mathbf{u}}_P^{rp}$ es el vector posición de un punto del perfil del carril en el sistema de referencia de la sección transversal. L_r es la distancia ideal del sistema de referencia de la vía y el sistema de referencia de cada carril.

c

Las matrices de transformación $\mathbf{A}^{t,rp}$ de los carriles izquierdo y derecho se pueden obtener como:

$$\mathbf{A}_{left}^{t,rp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta+\delta) & -sin(\beta+\delta) \\ 0 & sin(\beta+\delta) & cos(\beta+\delta) \end{bmatrix}, \mathbf{A}_{right}^{t,rp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\beta+\delta) & -sin(-\beta+\delta) \\ 0 & sin(-\beta+\delta) & cos(-\beta+\delta) \end{bmatrix} \tag{2.8}$$

donde el ángulo β es el ángulo que, por construcción, se le da a los carriles con respecto a la horizontal (peralte) mientras que el ángulo $\delta = [(\mathbf{r}_{left}^{irr})_z, (\mathbf{r}_{right}^{irr})_z]/2L_r$ es debido a la irregularidad.

Finalmente el vector $\overline{\mathbf{u}}_P^{rp}$ viene dado por la curva del perfil del carril:

$$\overline{\mathbf{u}}_P^{rp} = [0, s_2^r, f^{rp}(s_2^r)]^T \tag{2.9}$$

donde s_2^r es el parámetro transversal del perfil del carril y f^{rp} es la función que da lugar a la curva del perfil en función de dicho parámetro.

Como puede observarse las matrices 2.8 son matrices de rotación respecto al eje x (ver apéndice A).

2.3. Cinemática de la visión artificial desde un vehículo en movimiento

El algoritmo de auscultación y respuesta dinámica del vehículo pretende determinar las cuatro componentes de la irregularidad de la vía, es decir,las cuatro componentes no nulas de los vectores $\mathbf{r}^{irr}_{left}$ y $\mathbf{r}^{irr}_{right}$, y las seis coordenadas de posición y orientación del vehículo ($\mathbf{q}^{cb} = [0, \mathbf{r}^{cb}_y, \mathbf{r}^{cb}_z, \varphi^{cb}, \theta^{cb}, \psi^{cb}]^T$) (9 incógnitas) en un conjunto de instantes durante el movimiento del vehículo en la vía. Para ello, hay que plantear y resolver un sistema de ecuaciones algebraico. Las ecuaciones se basan en que la posición de los puntos grabados e iluminados por el láser responden a la ecuación 2.5 y también a la ecuación 2.7. Además se cuenta con la posición de los puntos en la imagen y la posición de los puntos como pertenecientes al plano del láser. Como se van a utilizar imágenes filmadas por dos cámaras (una del carril izquierdo y otra del derecho) resulta el siguiente sistema de ecuaciones:

$$
\begin{aligned}
\mathbf{r}^{cb} + \mathbf{A}^{t,cb}\left[\overline{\mathbf{u}}^{cam}_{left} + \mathbf{A}^{cb,cam}_{left}\mathbf{v}^{cam}_{P,left}\right] &= \mathbf{r}^{rp}_{0,left} + \mathbf{r}^{irr}_{left} + \mathbf{A}^{t,rp}_{left}\overline{\mathbf{u}}^{rp}_{P} \\
\frac{f}{(\mathbf{v}^{cam}_{P,left})_z}\mathbf{v}^{cam}_{P,left} &= \mathbf{v}^{im}_{P',left} \\
\triangle^T_{left}\mathbf{v}^{cam}_{P,left} - d_{left} &= 0 \\
\mathbf{r}^{cb} + \mathbf{A}^{t,cb}\left[\overline{\mathbf{u}}^{cam}_{right} + \mathbf{A}^{cb,cam}_{right}\mathbf{v}^{cam}_{P,right}\right] &= \mathbf{r}^{rp}_{0,right} + \mathbf{r}^{irr}_{right} + \mathbf{A}^{t,rp}_{right}\overline{\mathbf{u}}^{rp}_{P} \\
\frac{f}{(\mathbf{v}^{cam}_{P,right})_z}\mathbf{v}^{cam}_{P,right} &= \mathbf{v}^{im}_{P',right} \\
\triangle^T_{right}\mathbf{v}^{cam}_{P,right} - d_{right} &= 0
\end{aligned}
\tag{2.10}
$$

Se trata de un conjunto de doce ecuaciones para calcular quince incógnitas (indeterminado). Estas incógnitas son las nueve mencionadas (cinco coordenadas del vehículo y cuatro irregularidades de los carriles) y las seis componentes de los dos vectores $\mathbf{v}^{cam}_{P,left}$ y $\mathbf{v}^{cam}_{P,right}$. Estas ecuaciones dependen de un total de veintiún parámetros: las coordenadas de posición y orientación de cada cámara (doce en total) que permiten obtener $\overline{\mathbf{u}}^{cam}_{left}$, $\mathbf{A}^{cb,cam}_{left}$, $\overline{\mathbf{u}}^{cam}_{right}$, $\mathbf{A}^{cb,cam}_{right}$, los cuatro parámetros de cada plano deláser (ocho en total), que permiten obtener $\triangle_{left}$, d_{left}, $\triangle_{right}$, d_{right}, y la distancia focal f. Los valores numéricos de los parámetros deben ser obtenidos por un proceso de calibración del equipo instalado. Además, para el planteamiento y solución de las ecuaciones 2.3 hay que extraer de las imágenes grabadas las cuatro componentes no nulas de los vectores $\mathbf{v}^{im}_{P',left}$ y $\mathbf{v}^{im}_{P',right}$.

Figura 2.4: Fotograma del perfil de la cabeza del carril

La resolución de este problema será posible porque no sólo disponemos datos de los sistemas de visión, sino que disponemos información adicional de los sensores inerciales, los sensores que miden la distancia entre el *car body* y el *bogie*, y del tacómetro, encargado de medir la velocidad y posición de las ruedas del vehículo.

2.3.1. Identificación de un punto en el perfil del carril

Las ecuaciones cinemáticas planteadas son válidas siempre que seamos capaces de observar la posición de un único punto P de referencia en el perfil del carril, es decir, un punto con una posición concreta s_2^r con respecto al sistema de referencia del carril (ver ecuación 2.9).

En la figura 2.4 podemos ver un ejemplo de una imagen grabada por una de las cámaras. Se trata de una imagen en escala de grises (monócroma) que procesada mediante técnicas de visión artificial puede obtenerse una imagen binaria con una curva de un sólo píxel de espesor, dicho de otra manera, un vector en el que cada elemento sea la altura y de un píxel del perfil y el índice del elemento en el vector la posición x de dicho píxel en la imagen.

2.3.1.1. Segmentación del perfil del carril iluminado por el láser en una imagen en escala de grises

Una imagen monócroma es sólo una matriz donde cada elemento es el valor de intensidad de su correspondiente píxel. Como se ha mencionado antes, es necesario extraer la información de la curva del perfil del carril de la imagen antes de poder calcular la posición de cada uno de sus puntos respecto

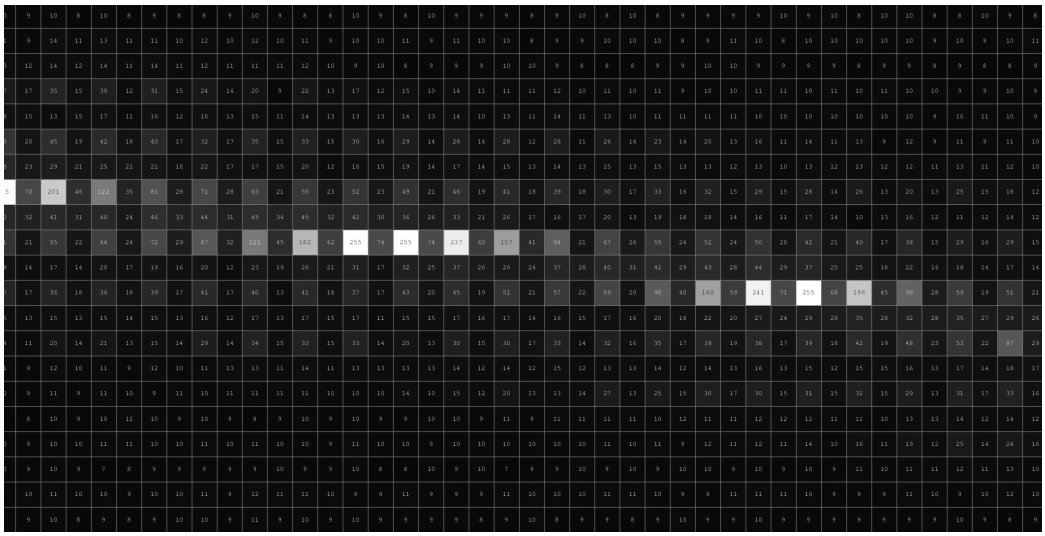

Figura 2.5: *Zoom* en una pequeña sección de la imagen del perfil del carril

al sistema de referencia de la cámara, es decir, es necesario un proceso de segmentación. A continuación, se presenta una forma simple de hacer esto con una precisión aceptable.

En la figura 2.5 se ha hecho *zoom* en una sección rectangular de la imagen del perfil del carril. Por dicha sección pasa la curva de la nube de puntos iluminados por el láser lineal. Estos son los puntos más claros. Cada cuadro es un píxel de la imagen y podemos ver en cada uno de ellos su valor de intensidad (es necesario ampliar la imagen), valores que varían entre 0 -negro- y 255 -blanco-. Como se puede observar en la imagen, esta nube de puntos no es un conjunto de píxeles con un valor común, sino que varían entre valores muy claros (255) de algunos de los píxeles interiores y se van volviendo más oscuros a medida que se alejan. Cuando se está lo suficientemente lejos de la curva los píxeles toman valores menores de 12. Sin embargo, el ruido introducido por la cámara hace que existan píxeles aislados con valores cercanos a 20.

El primer paso es decidir a partir de que valor se considera a un píxel perteneciente a la nube de la curva, es decir, hay que umbralizar la imagen. En la figura 2.6 se puede observar la imagen binaria resultante utilizando un valor de umbral de intensidad 25.

Una vez binarizada la imagen, se debe encontrar la curva de espesor 1 que está centrada en la nube de puntos, como la curva roja en la figura 2.7.

Una forma simple computacionalmente rápida para hacer esto se basa la suposición de que un gran porcentaje de los puntos del perfil del carril iluminados por el láser pertenecen a la parte superior del perfil que tiene una

Figura 2.6: Imagen del perfil del carril umbralizada

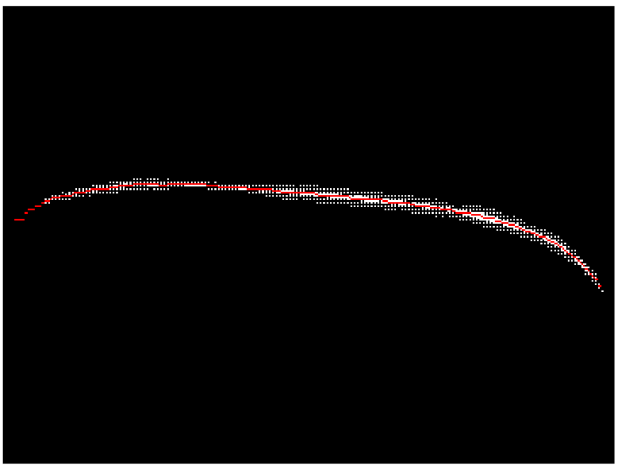

Figura 2.7: Curva media de la nube de puntos

curvatura muy pequeña y que el perfil del carril gira pequeños ángulos respecto al sistema de referencia de la cámara. Suponiendo esto, puede buscarse el punto medio de los píxeles de cada columna. Las columnas cuyo valor medio sea 0 (no existen píxeles iluminados en dicha columna) se considera que no pertenecen al perfil y no son tenidos en cuenta por el posterior algoritmo de identificación de la posición del perfil. Los resultados de la figura 2.7 han sido calculados utilizando este simple cálculo.

Los resultados son satisfactorios excepto en los extremos de la nube de puntos dónde la curvatura es mayor (los laterales de los perfiles ferroviarios son curvas correspondientes a circunferencias de pequeño radio, ver figura 2.8). El uso de métodos más sofisticados de visión artificial con más operaciones o basados en el color (para lo que sería necesaria una imagen en color) requerirían de sistemas electrónicos con procesadores muy potentes o el uso de FPGA que puedan hacer cálculos de forma paralela.

Un vector de puntos no es suficente para resolver el problema cinemático. Para conocer la posición y orientación del perfil del carril respecto al sistema de referencia de la cámara es necesario conocer, al menos, la posición de dos puntos característicos de esta. Por punto carácterístico se entiende un punto de interés desde el punto de vista de su cálculo analítico (por ejemplo un punto de inflexión). Con uno de los puntos característicos se puede fijar la posición del perfil en ese punto. Sin embargo, seguirían existiendo infinitas soluciones para la orientación del perfil. Conociendo un segundo punto del entonces se puede fijar también la orientación del perfil.

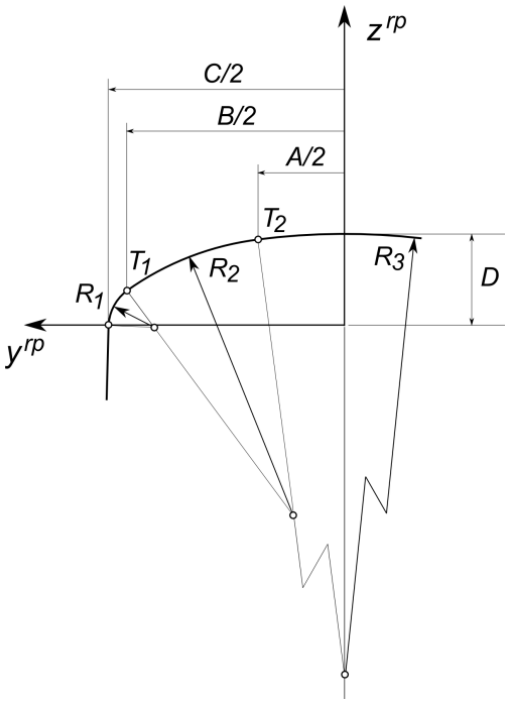

Figura 2.8: Perfil de la cabeza del carril tipo UIC 54 E1 [5]

Se ha propuesto una forma de calcular estos dos datos, aprovechando el conocimiento de la geometría del perfil comunmente utilizado en vías ferroviarias, mostrado en la figura 2.8, concretamente el UIC 54 E1. El perfil está formado por arcos de circunferencia de diferentes radios. Además se conocen los radios de dichos circulos y los puntos de tangencia entre las circunferencias. Este conocimiento de la geometría, de la vía junto a la suposición de que el sistema de referencia de la cámara y del perfil del carril tienen una orientación muy similar (esto quiere decir que el perfil proyectado en la imagen no está deformado) van a ser utilizados para conocer la posición y orientación del perfil.

2.3.1.2. Ajuste con dos circunferencias

El método propuesto se basa en buscar la curva de mejor ajuste a dos de las circunferencias que forman el carril por el método de minimización de mínimos cuadrados.

Como puede observarse en la figura 2.7 está formado por arcos de circunferencia de diferentes radios. El hecho de que el perfil del carril esté formado por arcos de circunferencia de diferentes radios va a ser utilizado a continua-

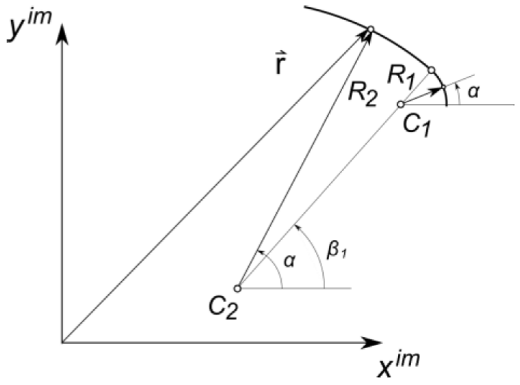

Figura 2.9: Arcos de circunferencia del perfil de la cabeza del carril tipo UIC 54 E1 [5]

ción para identificar uno de los puntos de tangencia de dichas circunferencias en el perfildel carril.

Si el perfil se observa en la imagen sin deformación, entonces se conoce la expresión matemática de la curva paramétrica del perfil. La figura 2.9 muestra un segmento del perfil que contiene al punto de tangencia T_1 entre las circunferencias con radios R_1 y R_2 (ver figura 2.8). Utilizando como parámetro de la curva el ángulo α que se observa en la figura 2.9, la expresión de las coordenadas de un punto cualquiera de la curva resulta (respecto al sistema de referencia del perfil del carril):

$$f(n) = \begin{cases} \mathbf{r}_{C1} + R_1[\cos(\alpha), \sin(\alpha)]^T, & \text{si } \alpha < \beta_1 \\ \mathbf{r}_{C2} + R2[\cos(\alpha), \sin(\alpha)]^T, & \text{si } \alpha > \beta_1 \end{cases} \tag{2.11}$$

donde los vectores posición de los centros C_1 y C_2 y el ángulo β_1 vienen dados por:

$$\mathbf{r}_{C1} = [x_{C1}, y_{C1}]^T, \mathbf{r}_{C12} = [x_{C2}, y_{C2}]^T, \beta_1 = \arctan\left(\frac{y_{C1} - y_{C2}}{x_{C1} - x_{C2}}\right) \tag{2.12}$$

Como se muestra en la figura el resultado del primer procesado de la imagen es una nube de puntos como los mostrados en la figura 2.10. Esta nube de puntos se puede ajustar con el procedimiento de los mínimos cuadrados a una curva del tipo de la ecuación 2.11. Para ello se minimiza la distancia cuadrática de los puntos de la nube a los puntos pertenecientes a una curva que responde a la ecuación 2.11. Esta distancia cuadrática se calcula como:

$$d^2 = [\hat{\mathbf{r}}_i - \mathbf{r}(\alpha_i)]^T[\hat{\mathbf{r}}_i - \mathbf{r}(\alpha_i)] \tag{2.13}$$

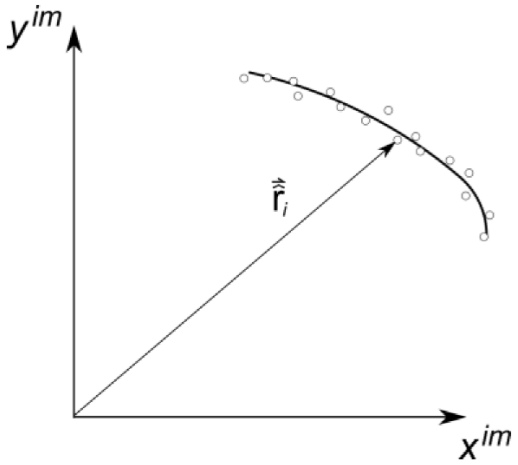

Figura 2.10: Puntos del perfil identificados mediante visión artificial [5]

Siendo $\hat{\mathbf{r}}_i$ el vector posción en el plano del punto i de la nube. En esta ecuación el ángulo α_i resulta:

$$\alpha_i = \begin{cases} \arctan\left(\frac{y_i - y_{C1}}{x_i - x_{C1}}\right), & \text{si } \alpha_i < \beta_1 \\ \arctan\left(\frac{y_i - y_{C1}}{x_i - x_{C1}}\right), & \text{si } \alpha_i > \beta_1 \end{cases} \tag{2.14}$$

Por tanto, el ajuste de la curva por mínimos cuadrados resulta de minimizar la función:

$$f(\mathbf{z}) = \sum_{i=1}^{np} [\hat{\mathbf{r}}_i - \mathbf{r}(\alpha_i)]^T [\hat{\mathbf{r}}_i - \mathbf{r}(\alpha_i)]$$
$$\mathbf{z} = [x_{C1}, y_{C1}, x_{C2}, y_{C2}]^T \tag{2.15}$$

siendo np el número de puntos de la nube considerados. La ecuación 2.15 es un problema de minimización con restricciones. Es necesario imponer mediante una ecuación algebraica de restricción que la distancia entre los puntos C_1 y C_2 coincide con la diferencia de radios R_2 y R_1. Esta ecuación resulta:

$$g(\mathbf{z}) = (x_{C1} - x_{C2})^2 + (y_{C1} - y_{C2})^2 - (R_2 - R_1)^2 = 0 \tag{2.16}$$

Por tanto, el ajuste de la curva del perfil aparece como resultado de resolver el problema:

$$\begin{aligned} &\min\ f(\mathbf{z}) \\ &\text{sujeto a } g(\mathbf{z}) = 0 \end{aligned} \tag{2.17}$$

Utilizando el método de los Multiplizadores de Lagrange, este problema es equivalente a la resolución del siguiente sistema de 5 ecuaciones algebraicas no-lineales:

$$\begin{aligned} \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} + \lambda \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} &= 0 \\ g(\mathbf{z}) &= 0 \end{aligned} \tag{2.18}$$

donde las incógnitas son las cuatro componentes de $\mathbf{z}$ y el Mutiplicador de Lagrange λ. Por tanto, el ajuste de la curva del perfil se resuelve con un sistema de cinco ecuaciones algebraicas no-lineales cuyas incógnitas son $\mathbf{z} = [x_{C1}, y_{C1}, x_{C2}, y_{C2}]^T$ y λ.

Una vez ajustado el perfil se puede identificar en este, por ejemplo, el punto de tangencia T_1 de los perfiles izquierdo y derecho del lado externo de la cabeza del carril. De entre los dos puntos T_1 que hay en cada carril (por simetría de la sección) es conveniennte tomar como referencia el del lado externo a la vía porque la sección del carril se suele desgastar por su lado interno pero no por el externo debido al rozamiento con la rueda.

Para hacer posible la identificación de un punto del perfil como se muestra en esta sección hay que buscar solución a la dificultad de que el perfil no se observa indeformado en la imagen. Esta dificultad se puede salvar con un método aproximado porque la orientación de la cámara y la del láser con respecto a la vía oscilan ligeramente (por estar montados en un vehículo en movimiento) respecto a la vía.

Capítulo 3

Equipo de sensores

Los sensores son dispositivos que nos van a permitir convertir la información del mundo real en datos tratables por un sistema electrónico.

Se han utilizado los siguientes sensores:

a. **Sistema de percepción**. Se ha colocado un sistema de percepción por cada carril. Este sistema de percepción está compuesto cada uno por una cámara de vídeo y un láser de proyección lineal. Cada sistema va colocado en el *car body* en cada uno de los laterales para ser capaz de grabar la posición relativa de los carriles así como el desgaste de su perfil. Además se ha colocado una tercera cámara de vídeo en la parte central del *car body* para poder filmar el paso de las traviesas y otros elementos de la vía.

b. **IMU (Inertial Measurement Unit)**. Se han utilizado tres sensores inerciales en el vagón instrumentado. Uno de ellos situado en en el centro de la cabina y los dos restantes en cada uno de los lados del *bogie*, justo en la parte superior de los amortiguadores. Además se ha colocado un sensor

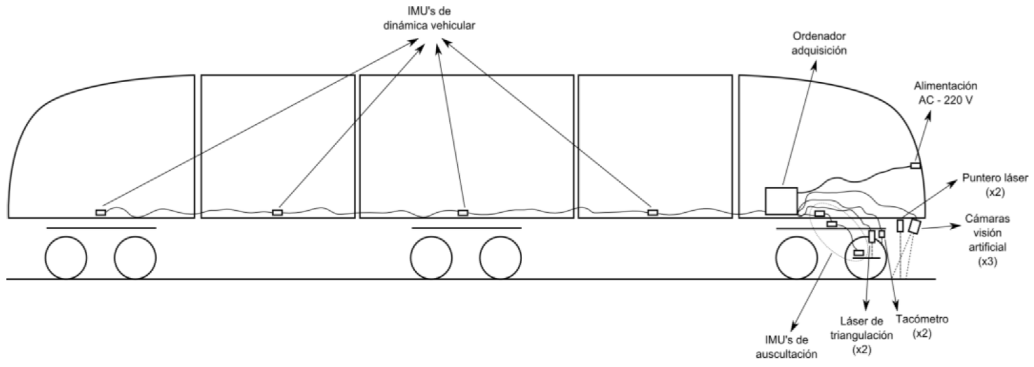

Figura 3.1: Vehículo instrumentado [5]

inercial en cada uno de los vagones restantes (situados en la cabina) con el objetivo de conocer la influencia de la dinámica de un vagón en el resto del vehículo. Cada uno de estos sensores está orientado de forma que su eje x quede alineado con el eje longitudinal (dirección de avance del vehículo).

c. **Sensor de distancia**. Se ha colocado un sensor de distancia perpendicular en cada lateral del *car body* midiendo la distancia entre este y el *bogie*.

d. **Sensor magnético**. Se ha colocado un sensor magnético para dos de las ruedas del vagón, una de cada lado, con el objetivo de medir su velocidad y posición (tacómetro).

En la figura 3.1 puede observarse la posición de cada uno de los sensores.

En este capítulo se van a describir las características de los distintos sensores utilizados en el proyecto y se va a justificar su uso.

3.1. Justificación del equipo de sensores

Este proyecto es fundamentalmente experimental, tanto desde el punto de vista del sistema de adquisición cómo de los modelos matemáticos involucrados. Es por ello que el número de sensores es mayor que el estrictamente necesario para la resolución del problema.

El resultado es un sistema redudante. El uso de un sistema redundante es muy útil en proyectos experimentales por los posibles fallos que puedan ocurrir y para tener distintas fuentes de datos que ayuden a validar los modelos matemáticos.

En la sección 1.1 se mencionó la importancia de conocer la longitud de onda de las irregularidades en el diseño de un sistema de adquisición de datos para la auscultación de vías. Efectivamente, la frecuencia de adquisición de los sensores está íntimamente ligada a la longitud de onda de las irregularidades.

La frecuencia característica (en Hz) de las irregularidades depende de su longitud de onda característica y de la velocidad del vehículo ferroviario:

$$f = \frac{v}{\lambda} \tag{3.1}$$

donde v esla velocidad que puede alcanzar el vehículo en m/s y λ la longitud de onda característica de las irregularidades en m.

La frecuencia de adquisición de los sensores debe ser, al menos, diez veces la frecuencia característica para ser capaz de reconstruir las irregularidades de la vía. En el caso de los ensayos realizados, el vehículo era capaz de alcanzar los 19,44 m/s (70 km/h) con una longitud de onda característica de 1 m

(ver sección 1.1) se calcula una frecuencia característica de 20 Hz. Como la frecuencia de adquisición debe ser diez veces mayor que la frecuencia característica, se admitirá una frecuencia de adquisición mínima de 200 Hz para todos los sensores involucrados en la reconstrucción de las irregularidades.

3.2. Sistema de percepción

Reconstruir un carril desde un vehículo ferroviario no es una tarea simple. El vehículo puede alcanzar grandes velocidades y por ello el sistema de percepción tiene que ser capaz resolver la problemática asociada:

1. El sistema tiene que tomar la información del perfil del carril a alta velocidad para que todos los puntos detectados por el sistema deben tener la misma coordenada longitudinal.

2. La frecuencia de escaneo debe ser lo suficientemente alta, lo suficiente como para identificar con precisión.

3. Aunque la posición del sistema de percepción es fija respecto al sistema de referencia del *bogie*, las vibraciones del vehículo pueden ser suficentes como para que haya un desplazamiento entre el sistema y el *bogie* no despreciables. Estas vibraciones pueden añadir una gran complejidad al modelo dinámico que reconstruye la posición y orientación del vehículo. Por esta razón la sujeción del sistema de percepción debe ser muy robusto y el sistema debe pesar lo menos posible.

4. Al pasar el carril por debajo del vehículo, se reducen las posibilidades de posiciones donde montar el sistema: en los laterales del vehículo, en las caras delantera o trasera, o en la cara inferior del vehículo. Cualquier montaje que sobresalga de los límites del vehículo no es deseable por seguridad y otras razones. Por tanto, el sistema deberá montarse en la parte inferior. El poco espacio entre la cara inferior del vehículo y el suelo exige un equipo poco voluminoso.

3.2.1. Cámara de video

Una apuesta segura para este cometido son los sensores de distancia de barrido. Estos sensores devuelven la distancia hasta los distintos puntos en su ángulo de trabajo. Sin embargo estos sensores tiene un coste económico muy elevado.

Por ello se ha decidido utilizar una cámara de video y un láser de proyección lineal para la reconstrucción del perfil que colocadas como en la figura

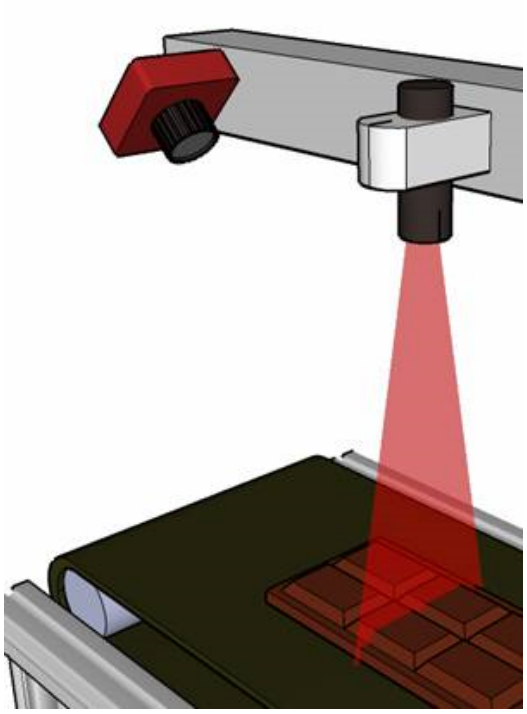

Figura 3.2: Reconstrucción con cámara y láser de proyección lineal [4]

3.2 y aplicando las ecuaciones 2.2 y 2.4 puede reconstruirse el perfil como puede observarse en la figura 3.3.

Si el número de fotogramas es lo suficientemente alto puede hacerse una reconstrucción 3D como puede verse en la figura 3.4.

La cámara elegida para esta función es la MQ003CG-CM de XIMEA. Se trata de una cámara CMOS a color con una resolución de 648x488 píxeles. Es una cámara muy pequeña (26x26x25 mm), con un peso de sólo 26 gramos y con la capacidad de hacer más de 500 fotogramas por segundo (FPS en adelante).

La cámara se comunica mediante una conexión USB (3.0). Dado el enorme flujo de datos transmitido por la cámara el fabricante recomienda conectar una sola cámara por tarjeta USB.

Para recibir y procesar las imágenes, el fabricante ofrece *xiAPI*, una completa libreria de funciones para la configuración, control y lectura de imágenes de la cámara desde nuestra propia aplicación. Esta API es además compatible con algunas de las más famosas librerías de visión artificial y procesado de imágenes como **OpenCV**.

También podemos usar programas comerciales para visualizar, guardar y procesar vídeos. En los ensayos realizados se ha utilizado un sotfware propor-

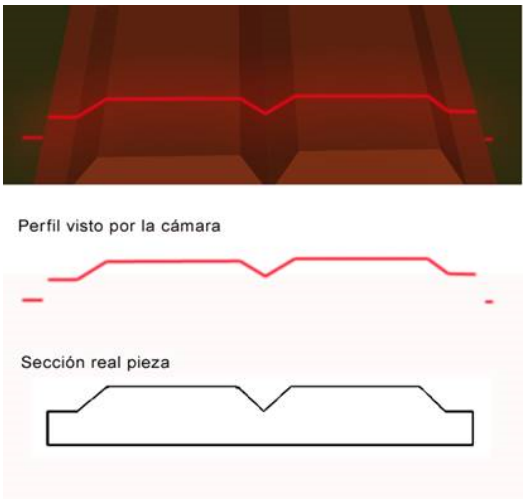

Figura 3.3: Reconstrucción con cámara y láser de proyección lineal [4]

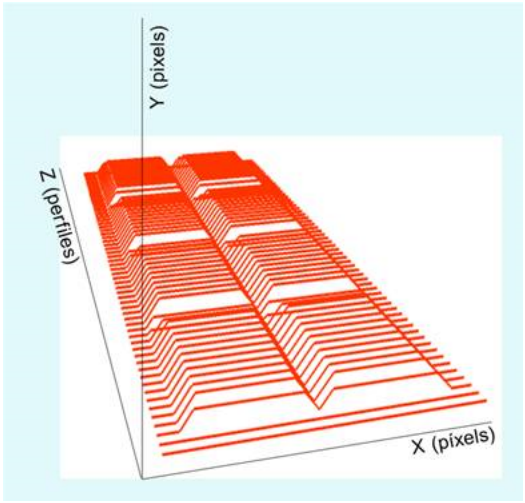

Figura 3.4: Reconstrucción 3D con varios perfiles [4]

cionado por el fabricante, *xiCamTool*. Este software, mostrado en la figura 3.6, permite visualizar y guardar vídeos e imágenes de las cámaras, además permite cambiar algunos parámetros de configuración determinantes en los ensayos:

1. Tipo y formato de la imagen. La cámara ofrece distintos formatos en los que transmitir la imagen, ya sea a color (RGB) o en escala de grises (monócrono). Para poder implementar directamente el proceso de segmentación utilizado en la sección 2.3.1.1 interesa utilizar un formato que devuelva una imagen en escala de grises. Los formatos en escala de grises disponibles son mono8, mono16, RAW8, RAW16. Los formatos de 16 bits tienen más calidad que los de 8 y debido a que envían dos bytes por píxel. El formato RAW16 es la imagen devuelta por la cámara sin ningún tipo de corrección. El formato mono16 hace hace un pequeño procesado para mejorar la imagen. Sin embargo, requiere de una CPU de mayor potencia que el formato RAW16. En las pruebas realizadas antes de los ensayos se pudo comprobar que el PC guardaba con dificultades (bajadas de FPS) vídeos a más de 30 FPS. Aunque este problema con los FPS probablemente tenga más relación con la velocidad a la que el PC puede guardar imágenes en el disco duro, que con la velocidad de recepción y procesado de las imágenes, se ha dedicido utilizar el formato RAW16 para no cargar con más trabajo a la CPU del PC.

2. Tiempo de exposición. El tiempo de exposición es el tiempo que se deja entrar a la luz en el sensor. Este parámetro es crítico en imágenes en movimiento debido a que si el tiempo de exposición es demasiado alto un mismo punto de un objeto puede proyectar luz en varios elementos del sensor de la cámara, por lo que se iluminarían varios píxeles en la imagen, viéndose el objeto deformado y difumminado. Es por esto que el tiempo de exposición debe ser ser lo menor posible para que la curva en la imagen sea lo más fina posible, como en la figura 2.4 en la que el tiempo de exposición era de 1 ms. En la figura 3.5 se observa el un perfil con mayor espesor debido a un tiempo de exposición de 31 ms.

3. Ganancia. La ganancia permite multiplicar el valor de intensidad de cada píxel por un valor, aumentando o disminuyendo el rango de valores de intensidad. En los ensayos se ha utilizado la máxima ganancia fundamentalmente por dos razones:

    a. Como se mostró en la sección 2.3.1.1 en la segmentación del perfil hay que elegir un valor umbral de intensidad para binarizar la imagen. Cuanto mayor sea el rango de valores de intensidad mejor funcionará

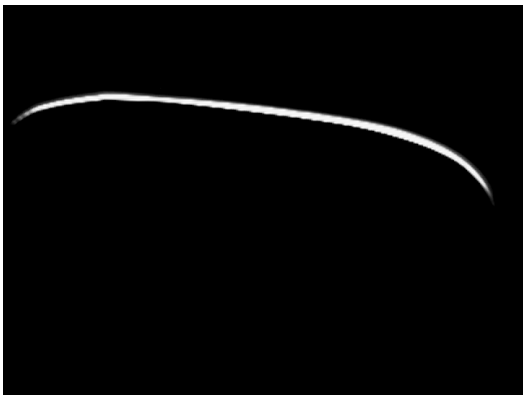

Figura 3.5: Fotograma del perfil de la cabeza del carril difuminado

el proceso de umbralización por la mayor diferencia entre valores claros (los puntos iluminados por el láser) y los oscuros, es decir, la imagen tendrá un mayor contraste.

b. Configurar un tiempo de exposición bajo también baja la luz que entra en el sensor, por lo que sin una ganancia lo suficientemente elevada no se distinguiría la curva del perfil iluminada por el láser del resto de la imagen.

4. Control de FPS. Existen dos opciones para controlar cuándo y cada cuánto hace la cámara una fotografía: por software y por hardware. La opción más simple por software es configurar la cámara para que haga un número concreto de fotogramas por segundo y la cámara automáticamente las transmita al PC a la frecuencia configurada. Sin embargo, es muy importante que las cámaras hagan la fotografía exactamente en el mismo instante, es decir, tienen que estar *sincronizadas*, sino cada cámara haría una fotografía a longitudes distintas. Una sincronización muy precisa es sencilla utilizando la vía hardware, simplemente con una señal cuadrada común a ambas cámaras. La vía hardware ha sido la elegida para sincronizar las cámaras en este proyecto y se describirá a continuación.

Para sincronizar vía hardware las cámaras se requiere utilizar su conector de pines GPIO (General Purpose Input/Output) mostrado en la figura 3.7. Este conector está formado por tres pines, uno de entrada uno de salida y tierra o común. El pin de salida puede ser utilizado para notificar acciones de la cámara a otro sistema electrónico. Sin embargo, en este proyecto se va a utilizar el pin de entrada para al sincronización. En este pin debe conectarse una señal digital cuyo valor alto corresponda a 24 V y el valor bajo a 0 v. Conectando una señal cuadrada a la frecuencia que deesamos que la cámara

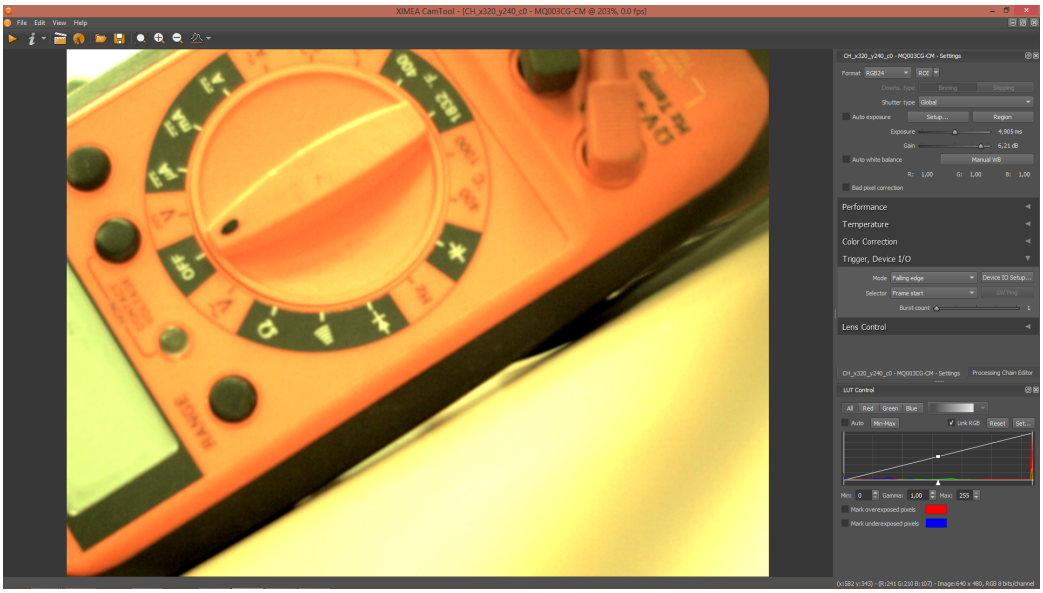

Figura 3.6: Software xiCamTool

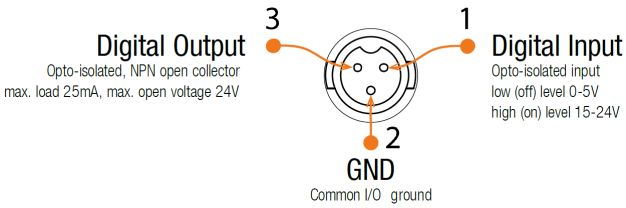

Figura 3.7: Pines GPIO de las cámaras [8]

haga una foto, la cámara la realizará en cada flanco de subida o de bajada (pueden configurarse ambas opciones) de la señal. Si esta señal es común para las dos cámaras del vehículo, ambas realizarán la fotografía exactamente en el mismo instante y estarán totalmente sincronizadas.

Para el control de la temporización de las cámaras, el sistema de adquisición de datos sólo tiene que crear y controlar la temporización de esta señal de sincronización.

3.2.2. Lente

La cámara de vídeo es una matriz de sensores ópticos con un sistema electrónico que controla, lee y transmite la información que llega a cada uno de los elementos de dicha matriz. El objetivo es generar fotogramas de un objeto (el perfil del carril iluminado por el láser de proyección lineal), y con

la mayor precisión posible, por lo que también es necesario que al sensor de la cámara llegue la luz proyectada por dicho objeto. Para ello necesitamos la óptica o lente.

La lente utilizada es la Fujinon HF12.5HA-1B, con una distancia focal de 12,5 *mm*. Esta lente fue recomendada por el fabricante para el enfoque correcto de objetos entre 30 y 45 *cms*, distancia existente entre la cámara y el perfil de la cabeza del carril iluminado por el láser. La lente permite un enfoque preciso de manera manual con una apertura relativa máxima de 1:1.4. También permite cambiar la apertura del iris con un rango F1.4-F16.

3.2.3. Láser de proyección lineal

El láser de proyección lineal nos va a permitir crear la iluminación estructurada. El láser de proyección lineal elegido tiene una potencia de 50 mW alimentado con una tensión de 5 V. Genera un haz pano láser a 120º desde su cabezal. El láser puede ser ajustado para conseguir una proyección nítida y de un espesor de aproximadamente 1 mm en objetos a distinta distancia (el rango es de varios metros).

3.2.4. Cámara de grabación del viaje

Al inicio del capítulo se enumeraron los sensores, entre ellos se encontraba una tercera cámara cuya función no es grabar la curva del perfil de los carriles. Esta tercera cámara es una cámara ajena a la auscultación, que ha sido colocada para grabar toda la vía al completo, tanto los carriles como las traviesas, como otros elementos que puedan encontrarse en el recorrido del vehículo.

Para cumplir este fin se ha elegido utilizar la Logitech HD Pro Webcam C920. Esta cámara con una resolución de 1920x1080 píxeles (FUll HD), enfoque automático y el uso del formato de compresión H264 es la solución perfecta para la grabación de vídeos con gran resolución sin necesidad de un alto FPS.

3.3. Sensor de distancia

El sensor de distancia permite conocer la distancia entre el *car body* y el *bogie* en cada uno de los laterales del vehículo. Las variaciones en esta medida no tomarán valores grandes (rango pequeño) pero si será necesaria una gran precisión por parte del sensor.

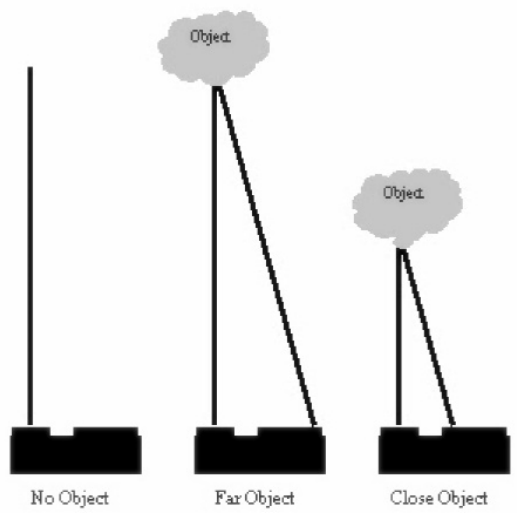

Figura 3.8: Funcionamiento del sensor de distancia por triangulación óptica [9]

Por ello se ha elegido el sensor de triangulación óptica *optoNCDT 1302*. Este sensor proyecta un punto de luz sobre la superficie objetivo, este es reflejado hacia un array de sensores ópticos. La posición del sensor óptico del array permite calcular la distancia relativa entre la superficie objetivo y el sensor como puede observarse en la figura 3.8.

El sensor tiene un rango de 200 mm, con una precisión de 100 μm y una frecuencia de medición y transmisión de las medidas de 750 Hz. Como puede observarse en la figura 3.9 el rango de 200 mm no comienza en el cabezal del sensor sino a una distancia SMR (Start of Measuring Range) desde este.

El sensor incluye un software propietario con el que se pueden configurar algunas características del modo de funcionamiento del sensor. En la figura 3.10 pueden observarse algunas de las posibilidades de la configuración del sensor.

El sensor proporciona dos vías hardware por las que transmitir la información de las medidas. Una de ellas es una salida analógica cuya corriente será proporcional a la distancia medida. La corriente se moverá en el rango 4 -20 mA siempre que la distancia medida esté dentro del rango del sensor. Cuando la superficie objetivo no está dentro del rango del sensor la corriente se fijará en 3,75 mA, como puede observarse en la figura 3.9. Para utiizar está vía de información debe conectarse el pin 11 (ver figura 3.11), no olvidar conectar el pin GND. Además el sensor debe tener conectado a GND el pin 8 para activarse.

Por otro lado existe un puerto digital por el que transmitir la información.

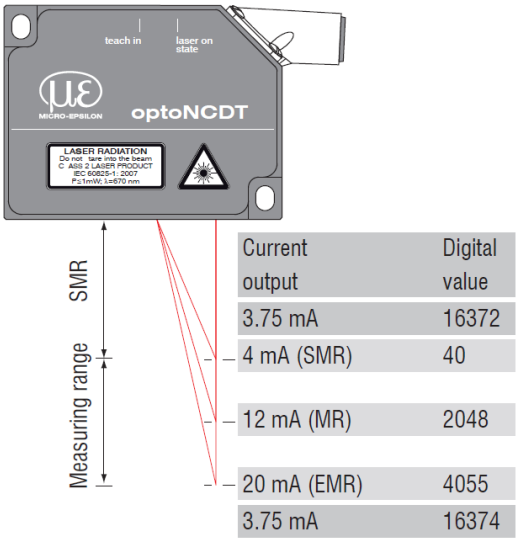

Figura 3.9: Salida del sensor de distancia [10]

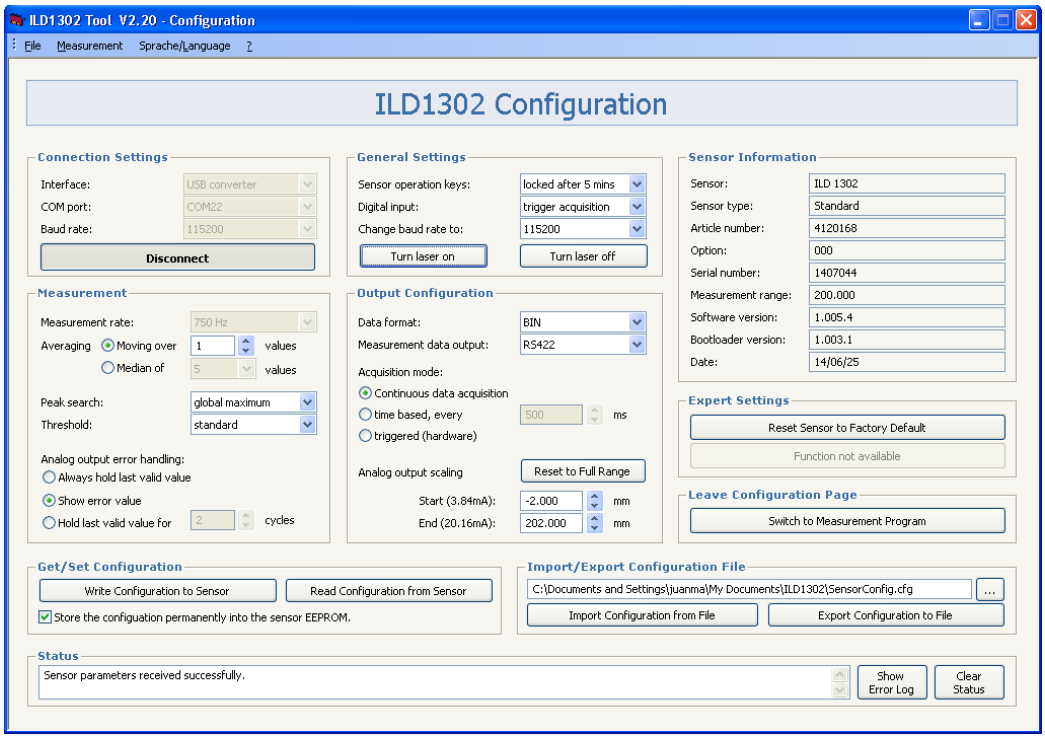

Figura 3.10: Configuración del sensor de distancia optoNCDT 1302

Pin	Description		Color code PC1402-x/I	Specification
3	RS422 Rx+	Serial input	green	Internally terminated with 120 Ohm
4	RS422 Rx-		yellow	
5	RS422 Tx+	Serial output	grey	Terminate externally with 120 Ohm
6	RS422 Tx-		pink	
7	$+U_B$		red	11 ... 30 VDC, typical 24 VDC / 50 mA
8	Laser off	Switch input	black	Laser is active, if pin 8 is connected with GND
9	Teach in		violet	Connected to GND for at least 30 ms
10	Error	Switch output	brown	Open-Collector (NPN), I_{max} = 100 mA, U_{max} = 30 VDC, short circuit proof, turn off the power supply to reset the short circuit protection
11	I_{OUT}	4 ... 20 mA	white	R_{Load} = 250 Ω results U_{OUT} 1 ... 5 V with U_B > 11 V R_{Load} = 500 Ω results U_{OUT} 1 ... 10 V with U_B > 17 V
12	GND		blue	Supply and signal ground
1/2	n.c.			

Figura 3.11: Pinout del sensor de distancia [10]

Se trata de una salida serie RS422 (pines 3-6, ver figura 3.11). Una vez conectado a una fuente de alimentación de 24 V el sensor comienza a hacer medidas y enviarlas por su puerto serie.

Existen distintas posibilidades de leer los datos de este sensor. Una es conectarlo directamente a otro dispositivo electrónico con un puerto serie compatible (como un microcontrolador, o un PC con este tipo de puerto). Sin embargo, también existe la posibilidad de usar un cable adaptador cuya salida es USB y así poder conectarlo a cualquier PC.

En caso de leer el sensor desde un PC, el sensor viene acompañado de MEDAQLib una API (Application Programing Interface) con librerias de funciones con las que podemos crear nuestra propia aplicación capaz de comunicarse con el sensor.

El sensor permite configurar dos formas de trasnmitir las distancias medidas. En la primera de ellas el paquete de datos codificado en binario contiene dos bytes (figura 3.12). Por cada byte enviado se envía un bit de inicio y uno que indica el final del byte, entre ambos los 8 bit del byte. En el primer byte se envía un bit a 1 (este bit puede usarse para identificar que byte estamos recibiendo) seguido de los 7 bit más significativos del valor medido. En el segundo byte se envía un 0 seguido de los 7 bit menos significativos del valor de la distancia medida (ver figura 3.13).

El sistema electrónico receptor debe reconstruir el paquete de forma que el orden final de los bit sea el que se observa en la figura 3.14. Esta sigue sin ser la distancia en mm buscada, sino que es un valor comprendido entre los valores 40-4055 proporcional a la distancia mientras el valor de distancia

Start	1	7 Bit MSB	Stop	Start	0	7 Bit LSB	Stop

Figura 3.12: Formato del paquete codificado en binario del sensor de distancia

H-Byte	1	D13	D12	D11	D10	D9	D8	D7
L-Byte	0	D6	D5	D4	D3	D2	D1	D0

Figura 3.13: Bytes del paquete codificado en binario del sensor de distancia [10]

medida está dentro del rango del sensor. Si la distancia medida es menor que la mínima el valor enviado por el sensor será 16372, en caso de sobrepasara el rango se enviará el valor 16374 como puede verse en la figura 3.9.

Para obtener el valor deseado en mm se debe hacer un pequeño cálculo:

$$\text{distancia}(mm) = \frac{rango(mm)}{rango_{max} - rango_{min}} V_r = \frac{200(mm)}{4055 - 40} V_r = 0,0498 V_r \tag{3.2}$$

donde V_r es el valor recibido por el sensor.

El sensor puede ser configurado para que envíe el valor caracter a caracter en formato ASCII. Sin embargo, esta forma de transmisión genera, evidentemente, un flujo de datos mayor que la codificación en binario por lo que no será utilizada en este proyecto.

Es importante conocer la temporización de este sensor. Tal y como puede deducirse de la figura 3.15 el sensor necesita cuatro ciclos de 1,3 ms para calcular la medida y transmitirla después del ciclo de exposición en el que se recibe la onda reflejada. Esto significa que el dato recibido se refiere a la distancia leída por el sensor $4x1,3ms = 5,2ms$ atrás en el tiempo. El conocimiento de este desfase temporal será fundamental para obtener unos resultados correctos y precisos.

3.4. *Inertial Measurement Unit* (IMU)

Las IMU serán las encargadas de proporcionar la mayoría de los datos necesarios para conocer la dinámica del vehículo.

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Figura 3.14: Valor enviado por el sensor de distancia reconstruido [10]

Cycle		1.	2.	3.	4.	5.	6.	
Time	max. 5 s	1.3 ms	2.6 ms	3.9 ms	5.2 ms	6.5 ms	7.8 ms	
	Initialisation including the output of the info string	Exposure N	Reading N	Computation N	Controlling N	Output N		
			Exposure N+1	Reading N+1	Computation N+1	Controlling N+1	Output N+1	
				Exposure N+2	Reading N+2	Computation N+2	Controlling N+2	...
		First exposure after power up of the sensor			Exposure N+3	Reading N+3	Computation N+3	...
						Exposure N+4	Reading N+4	...
							...	...

Figura 3.15: Tiempos en el proceso de adquisición del sensor de distancia [10]

Las IMU elegidas son las 3DM-GX4-25. Una de ellas, la que va colocada en la parte central del *bogie* del vehículo es la 3DM-GX4-45, que tiene la misma funcionalidad que la 3DM-GX4-25, añadiendo además GPS.

Estos sensores integran un acelerómetro, un giróscopo y un magnetómetro, todos ellos de 3 ejes.

El acelerómetro tiene un rango máximo de $\pm 16g$ y una resolución de $< 0,1mg$ con un ancho de banda máximo de 225 Hz.

El giróscopo tiene un rango máximo de $\pm 900^o/s$ y una resolución de $<0{,}008^o/s$ con un ancho de banda máximo de 250 Hz.

El magnetómetro no va a ser utilizado en este proyecto.

Tanto el acelerómetro como el giróscopo tiene una máxima frecuencia de envío de paquetes de datos de 1000 Hz.

Estos sensores tienen además la posibilidad de estimar medidas mediante la aplicación del Filtro de Kalman. Sin embargo, esta funcionalidad no va a ser utilizada en este proyecto.

Todas las características configurables del sensor pueden ser programas de forma sencilla gracias al software propietario, el MIP Monitor. En este proyecto se configurarán los datos que deseamos que transmita el sensor (aceleraciones lineales y velocidades angulares, ver figura 3.16) y la frecuencia de transmisión de los paquetes de datos y la velocidad de conexión del puerto UART (puerto serie *Universal Asynchronous Receiver-Transmitter*, ver figura 3.17). Para cambiar los datos enviados por el sensor y la frecuencia a la que este los transmite puede configurarse en Settings $->$ Device, en la pestaña IMU y Message Format. Para cambiar la velocidad de conexión del puerto UART, esta puede configurarse en Settings $->$ System.

El sensor viene acompañado de una libreria de funciones con las que poder configurar el sensor y leer datos con nuestras propias aplicaciones.

El sensor tiene dos puertos por los que comunicarse, ambos digitales. Una se trata de una salida USB y por otro lado también puede transmitir los datos

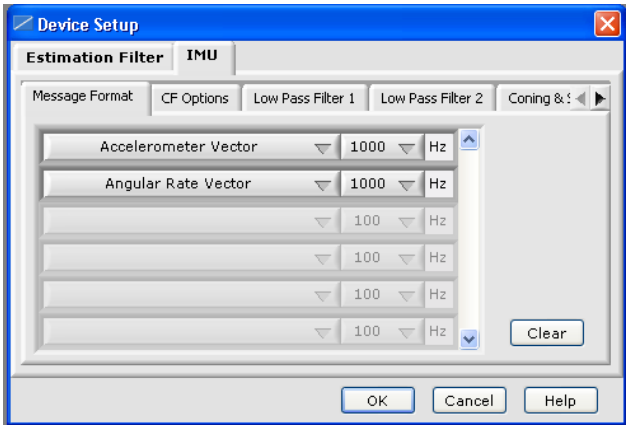

Figura 3.16: Configuración del formato del paquete de la IMU

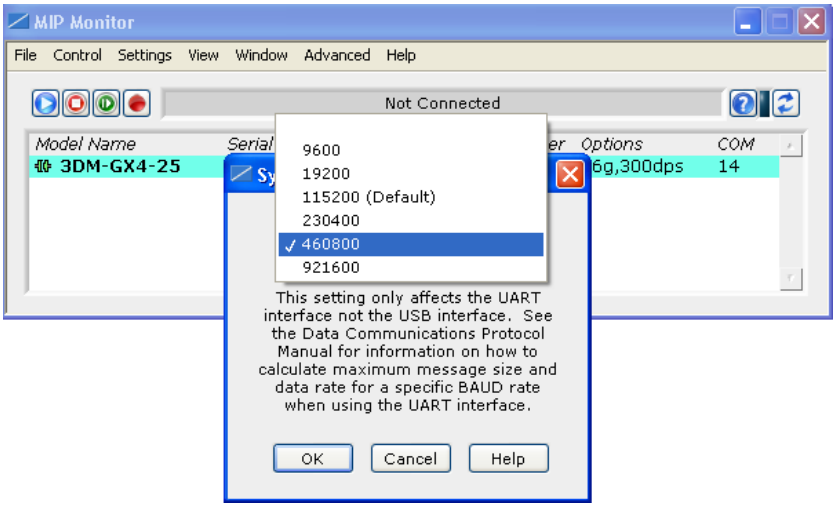

Figura 3.17: Configuración de la velocidad del puerto serie de la IMU

sensor connector

sensor pin number	signal
1	USB D-
2	USB D+
3 **	+3.2V to +5.5 VDC
4	RS232 RxD
5	RS232 TxD
6 **	+5.2V to +36 VDC
7	PPS input
8	GND
9	--

** use either pin 3 OR pin 6 for sensor power

Figura 3.18: Pinout del sensor 3DM-GX4-25 [12]

Header				Packet Payload			Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length byte*	*Field Length byte*	*Field Descriptor byte*	*Field Data: Accel vector (12 bytes, 3 float – X, Y, Z)*	*MSB*	*LSB*
0x75	**0x65**	**0x80**	**0x0E**	**0x0E**	**0x04**	**0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F**	**0x92**	**0xC0**

Figura 3.19: Paquete de datos de ejemplo 3DM-GX4-25 [11]

por su puerto serie RS232. Los pines de ambos puertos y de alimentación pueden verse en la figura 3.18. El pin de alimentación de 3,2 V - 5,5 V es utilizado cuando se transmiten datos por el puerto USB y el pin de 5,2 V - 36 V con el puerto serie RS232.

El sensor transmite los datos en paquetes codificados en binario. En la figura 3.19 se muestra un paquete de datos de ejemplo de la IMU en el que se está transmitiendo la aceleración en el los ejes cartesianos. Es posible diferenciar las diferentes partes del paquete, estas son *Header*, *Packet Payload* y *Checksum*. En la cabecera del paquete (*Header*) primero encontramos dos bytes que siempre son los mismos para todos los paquetes del sensor, que son 75 y 65 en hexadecimal (equivalen a los caracteres 'u' y 'e' en el código ASCII). Estos bytes pueden ser útiles a la hora de programar un algoritmo que detecte la llegada de un nuevo paquete.

El tercer byte es un descriptor del paquete, que en el ejemplo es 0x80, que en este caso indica que el paquete es un paquete de datos AHRS (Attitude and Heading Reference System). El cuarto byte de la cabecera indica el número de bytes de la siguiente parte del paquete *Packet Payload*, que en el ejemplo tiene una longitud de 14 bytes.

La siguiente parte del paquete es *Packet Payload*, que a su vez puede

Header				Packet Payload (2 fields)						Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descript or Set*	*Payload Length*	*Field1 Len*	*Field1 Descriptor*	*Field1 Data*	*Field2 Len*	*Field2 Descriptor*	*Field2 Data*	*MSB*	*LSB*
0x75	0x65	0x80	0x1C	0x0E	0x05	0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F	0x0E	0x06	0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F	0xB1	0x1E

Figura 3.20: Paquete de datos de ejemplo 3DM-GX4-25 con dos campos de datos [11]

estar dividido en varios campos (en la figura 3.20 podemos ver un paquete con dos campos). Cada campo contiene un dato del sensor (un campo podría contener las aceleraciones y otro las velocidades angulares). Como podemos observar en las figuras 3.19 y 3.20 cada campo tiene un primer byte que indica el tamaño del propio campo y un segundo byte que que describe el contenido de los datos del campo. Por último, nos encontramos un número variables de bytes que son los propios datos del campo.

La última parte del paquete *Checksum* son dos bytes que permiten comprobar si el paquete ha sido transmitido correctamente. Estos bytes son obtenidos al hacer operaciones binarias con los bytes del propio paquete. El sistema receptor puede hacer las mismas operaciones sobre los bytes del paquete recibido, en el caso de obtener los mismos valores del *Checksum* el paquete ha recibido contiene los mismos bytes que el enviado por lo que la transmisión ha sido correcta.

En el presente proyecto las IMU serán configuradas para enviar los datos de las aceleraciones y velocidades angulares (ver figura 3.16), por lo que el paquete de datos enviado tendrá el mismo formato que el de la figura 3.20. El número de bytes de estos paquetes es de 34, y los bytes contenedores de los datos son:

1. **Aceleración eje x:** Bytes 6 - 9
2. **Aceleración eje y:** Bytes 10 - 13
3. **Aceleración eje z:** Bytes 14 - 17
4. **Velocidad angular respecto al eje x:** Bytes 20 - 23
5. **Velocidad angular respecto al eje x:** Bytes 24 - 27
6. **Velocidad angular respecto al eje x:** Bytes 28 - 31

Como podemos observar cada valor de aceleración o velocidad angular viene dado por cuatro bytes. Estos bytes corresponden a un número real

según el estándar IEEE-754. El estándar IEEE-754 es el más extendido para representación de números en coma flotante.

Es importante también conocer el flujo de datos que va crear el sensor mandando dichos paquetes a una frecuencia de 1000 Hz. El sensor viene configurado de fábrica para cominicarse por su puerto serie a una velocidad de 115200 baudios (bits por segundo), pero puede ser que no sea suficiente.

Al ser una comunicación serie RS232, para cada byte, además de los 8 bit del propio byte, se envía un bit de inicio y uno de parada, que son en total 10 bit por byte enviado. Por lo que la mínima velocidad del puerto serie debe de ser de:

$$t_{min} = 10 N_{bytes} f \tag{3.3}$$

donde t_min es la velocidad mínima de configuración del puerto serie de la IMU para poder enviar todos los datos sin pérdidas, N_{bytes} el número de bytes del paquete de datos y f la frecuencia de envío de paquetes. En el caso de nuestro paquete de datos de 34 bytes, enviado a una frecuencia de 1000 Hz se calcula un valor de 34000 baudios, por tanto habrá que aumentar la velocidad de comunicación del puerto serie de la IMU (y la velocidad del puerto del dispositivo electrónico receptor). El puerto serie tiene unas velocidades concretas (ver figura 3.17) a las que se puede configurar, de todas ellas se ha elegido 460800 baudios por ser la menor (a mayor velocidad mayor probabilidad de errores o de incompatibilidades cuando las velocidades son altas) que supera el valor antes calculado.

La ecuación 3.3 es una ecuación típica para calcular la velocidad mínima de conexión en una comunicación serie asíncrona en ausencia de bit adicionales como un segundo bit de parada o el bit de paridad, por lo que se hará referencia a esta más adelante.

En el caso de leer el sensor desde una apicación en un PC utilizando la librería de funciones, ésta nos va a permitir abstraernos del protocolo de comunicaciones y del conocimiento del formato de los paquetes de datos y su reconstrucción.

Al inicio del presente capítulo se mencionó el uso de sensores inerciales en cada uno de los vagones del vehículo para registrar también la dinámica vehicular. Estos sensores son distintos a los usados en el vagón instrumentado para la auscultación de vías. Se trata de sensores inerciales *PhidgetSpatial* con acelerómetro, giróscopo y magnetómetro de tres ejes. Este sensor tiene peores carácterísticas que los de auscultación, una tarea de gran precisión, pero son ideales para recoger la dinámica vehicular.

El acelerómetro tiene un rango máximo de $\pm 5g$ y una resolución de $< 228 \mu g$ con un ancho de banda máximo de 110 Hz.

El giróscopo tiene un rango máximo de $\pm 400^o/s$ y una resolución de $<0{,}02^o/s$. El magnetómetro no va a ser utilizado en este proyecto.

Estos sensores vienen acompañados de un software propietario con el que leerlos y configurarlos, además el fabricante ofrece una extensa libreria de funciones para poder leer los sensores desde nuestra propia aplicación.

3.5. Sensor de campos magnéticos

Los sensores magnéticos van a ser utilizados para detectar el paso de imanes fijos a las ruedas del vehículo y así poder realizar la odometría del vehículo. Otras soluciones a este problema es utilizar sensores ópticos, como un transmisor receptor de infrarrojos y pegar pegatinas reflectantes en la rueda del vehículo, que puedan ser detectadas por el sensor óptico al reflejar las ondas del transmisor. Después de varias pruebas se decidió utilizar los sensores magnéticos por ser una forma más fiable y limpia de medir la velocidad de las ruedas (los imanes pueden ser simplemente separados de la rueda metálica, además la suciedad no es tan perjudicial como en el caso de los sensores ópticos).

Los sensores utilizados son sensores digitales. Para su correcto funcionamiento se deben conector sus tres pines. Dos de ellos son alimentación (3,3 V) y tierra. El tercero es el pin de señal, que se mantiene en nivel alto (3,3 V) cuando detecta un campo magnético cercano, como el creado por los imanes fijos a las ruedas metálicas.

Capítulo 4

Sistema de adquisición de datos

Un Sistema de Adquisición de Datos (SAD en adelante) es aquel dispositivo electrónico, que tomando la informacin del mundo real, la procesa hasta convertirla en datos en una memoria digital, un formato adecuado para el posterior tratamiento y presentación de la información.

En este proyecto la información del mundo real es la dinámica vehicular (aceleraciones, velocidades angulares, distancias entre partes) e imágenes proporcionadas por el sistema de visión artificial. Esta información tiene que ser enviada al PC, que la procesará para resolver el problema cinemático e identificar las irregularidades de la vía.

En este capítulo se van a describir los dos SAD propuestos para este proyecto. El primero de ellos consiste, básicamente, en una conexión directa entre el PC de procesamiento de datos y los sensores. La segunda opción incluye una etapa intermedia entre los sensores y el PC con microcontroladores que controlan el flujo de datos. Ambos sistemas buscan cumplir una serie de funcionalidades. Conocer estas funcionalidades es fundamental para entender su diseño, además permite comparar la eficiencia de ambos sistemas.

4.1. Funcionalidad del SAD

En esta sección se describen las características deseadas por un SAD en este proyecto.

a) **Inicialización y control de los sensores.**

Como vimos en el capítulo 3 es necesario enviarle a algunos sensores una señal (mensaje binario) de inicio para que comience a transmitir datos o incluso mantener una señal cuadrada (eléctrica) en algunas de las entradas del sensor para que este continue transmitiendo datos.

b) **Control de flujo de datos.**

Cada sensor utiliza un protocolo de comunicación distinto para transmitir los datos. Evidentemente, los paquetes de datos (mensajes binarios) enviados por cada sensor son de distinto tamaño y la reconstrucción de los datos se hace de forma de distinta. Además, la frecuencia de envío de paquetes de cada sensor es diferente. El SAD tiene que ser capaz de gestionar todos estos procesos sin pérdida de datos.

c) **Sincronización de sensores.**

El sistema embarcado tendrá que procesar la información proveniente de los sensores para obtener los resultados, para obtener unos resultados precisos el sistema debe tener un control riguroso del tiempo. Los sensores son a su vez un sistema independiente con su propio sistema de temporización. El SAD debe ser capaz de enviar al sistema de procesado cada uno de los paquetes de datos con una marca de tiempo respecto a una única referencia temporal: la que será utilizada para procesar los datos.

d) **Fiabilidad.**

Las comunicaciones en la electrónica son a menudo problemáticas. El ruido corrompe los paquetes de datos produciendo resultados incorrectos. Las malas conexiones también pueden ser el origen de paquetes de datos erróneos o discontinudidades en la frecuencia de envío de datos. El SAD debe ser diseñado de forma que sea robusto al ruido electrónico y debe estar preparado para un cambio de régimen de transmisión de datos.

Un fallo en el SAD puede ser fatal: en caso de utilizarse los resultados de la dinámica vehicular para actuar sobre el propio vehículo se podría provocar un mal funcionamiento del vehículo, un tramo de vía con irregularidades que no sea notificado peligroso para un vehículo ferroviario.

A continuación se presentan los dos sistemas propuestos cuyo objetivo será cumplir las funcionalidades mencionadas.

4.2. SAD integrado en el PC

Todos los sensores usados en el presente proyecto tienen conversores ADC (*Analog-to-Digital Converter*) con los que convertir las señales eléctricas continuas provenientes de los transductores, en señales digitales. Además, todos estos sensores contienen electrónica que permite una comunicación USB con otro sistema electrónico, ya sea de forma directa (cámaras y sensores inerciales) o mediante conversores externos (sensores de distancia). Esto permite

conectar los sensores a un PC de propósito general con suficientes puertos USB. La instalación de los driver y las API de los sensores en un PC con un sistema operativo compatible, como Windows, hace posible la creación de un aplicación de adquisición de datos, asbtrayéndose el programador de los complejos protocolos de comunicación.

Esta aplicación, diseñada y programada por *Daniel García Vallejo* [15], es capaz de leer los datos de los sensores y guardarlos en los archivos necesarios en el disco del PC, se puede ver su apariencia en las figuras 4.2 y 4.2. En la figura 4.1 se puede ver el esquema completo de conexiones y alimentación de todos los sesores en los ensayos. Todo el conjunto se alimenta de tensión alterna de 220 V del vehículo. Todos los sensores se alimentan directamente por su puerto de comunicaciones excepto los sensores de distancia. También es alimentado con 24 V el sistema *tacómetro*.

Las conexiones entre el PC y los sensores se realiza por puertos serie, excepto en el caso de las IMU *PhidgetSpatial*, que aunque también utilizan puertos serie, son conectados cada uno de ellas a una alargadera ethernet que permite llegar hasta cada uno de los vagones del vehículo. Los datos de las cámaras son recogidos por PC independientes utilizando el software proporcionado por el fabricante de las cámaras, el *xiCamTool* (sección 3.2.1). Este software sólo permite grabar imágenes de una cámara, además no pueden ejecuarse distintas instancias del programa en un solo PC, por esta razón, es necesario el uso de un PC para cada cámara.

La aplicación está preparada para funcionar en un sistema operativo windows. Para obtener el mayor rendimiento posible se ha diseñado el programa con distintos hilos que se ejecutan en paralelo:

a. Existe un hilo por cada sensor inercial de auscultación, en total tres, debido a la lenta velocidad de respuesta de estos. Los sensores son leídos mediante la técnica de *polling*, en la que se le manda al sensor una petición, el sensor la procesa y responde con los datos deseados. De esta forma, se consigue un periodo de adquisición de 6 ms (166 Hz), cada cierto tiempo se producen picos de 12 ms en el tiempo de adquisición.

b. Los dos sensores de distancia comparten un único hilo. Al igual que los sensores inerciales, son leídos por polling, obteniéndose un periodo de muestreo de 5 ms (200 Hz).

c. Existe un hilo común para los cuatro sensores inerciales que registran la dinámica de los vagones no instrumentados (las IMU *PhidgetSpatial*). Estos sensores son leídos por polling y se obtiene un periodo de adquisición de 8 ms (125 Hz).

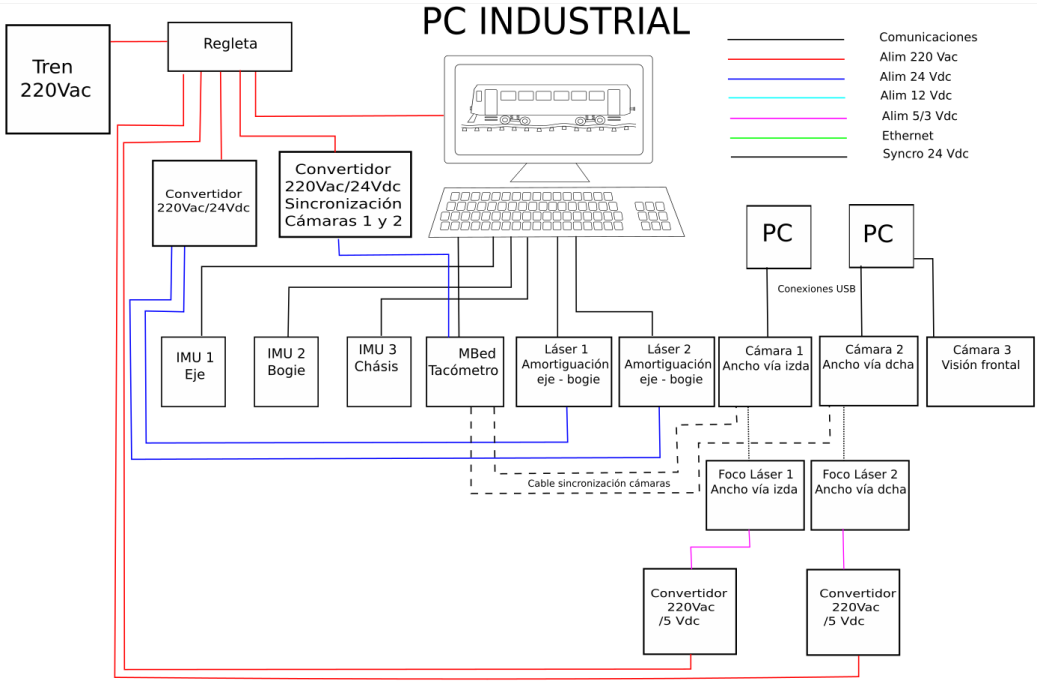

Figura 4.1: Arquitectura de conexiones y alimentación del SAD integrado en PC

Cada hilo tiene la función de leer los sensores correspondientes y guardar los datos en ficheros.

El programa tiene una referencia única de tiempo con precisión de microsegundos, el reloj del PC. Este reloj es único para todos los hilos, por lo que cada hilo sólo tiene que leer los datos de los sensores, y guardar el instante en el que se leyó el sensor además de la información enviada por el sensor, es decir, una marca de tiempo.

Existe un sensor utilizado en el proyecto que no contiene la electrónica necesaria para la comunicación con los puertos del PC. Se trata de los sensores magnéticos, vistos en la sección 3.5. Se ha desarrollado un sistema electrónico (sistema tacómetro en adelante) capaz de leer la información que proporciona este sensor y enviarla de manera ordenada por USB al PC utilizando un microcontrolador. Este sistema también implementa la función de crear la señal cuadrada de sincronización de las cámaras.

Este sistema va conectado por puerto serie USB al PC y es el único que no tiene un hilo asociado ejecutándose en paralelo. En su lugar, los bytes recibidos por el sistema tacómetro se guardan en un *buffer*. Dicho *buffer* va siendo leído cuando el programa no está ocupado con otra tarea y guarda los datos leídos en el fichero correspondiente.

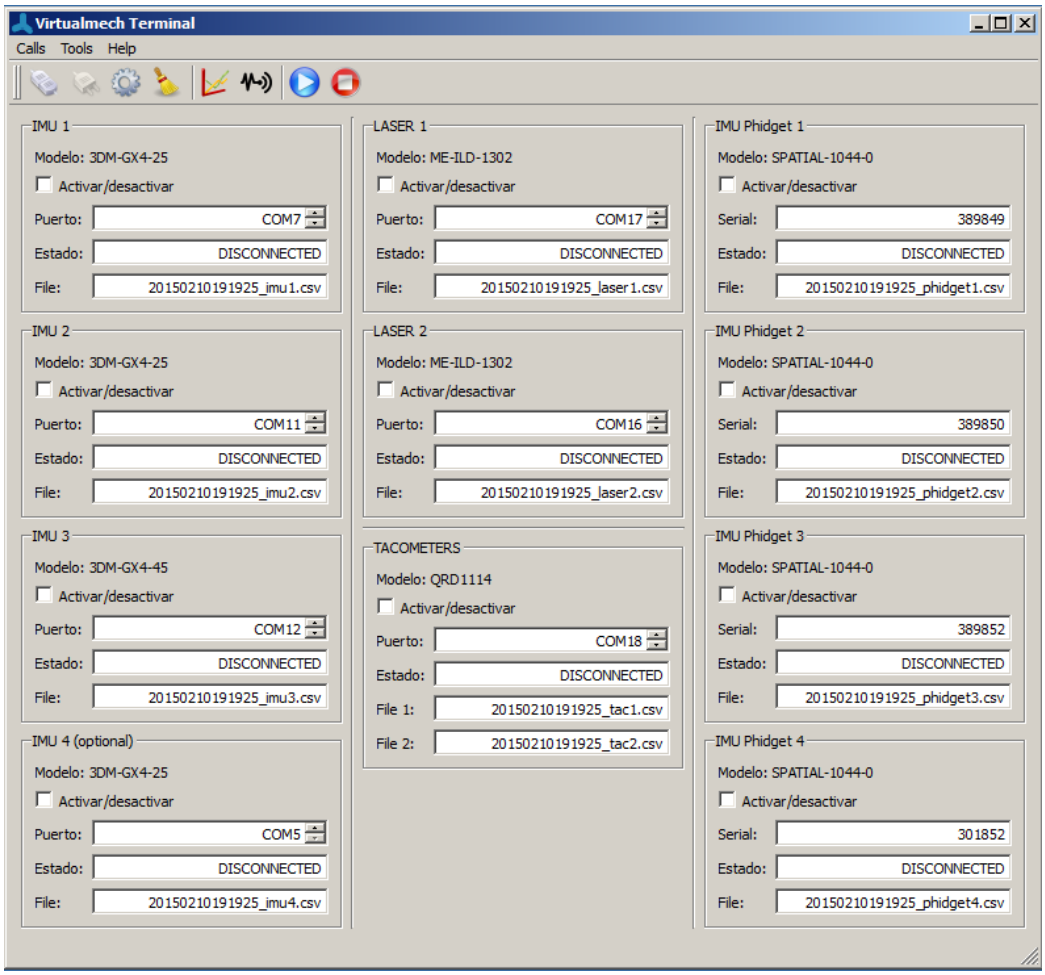

Figura 4.2: Panel de conexiones de la aplicación de adquisición de datos

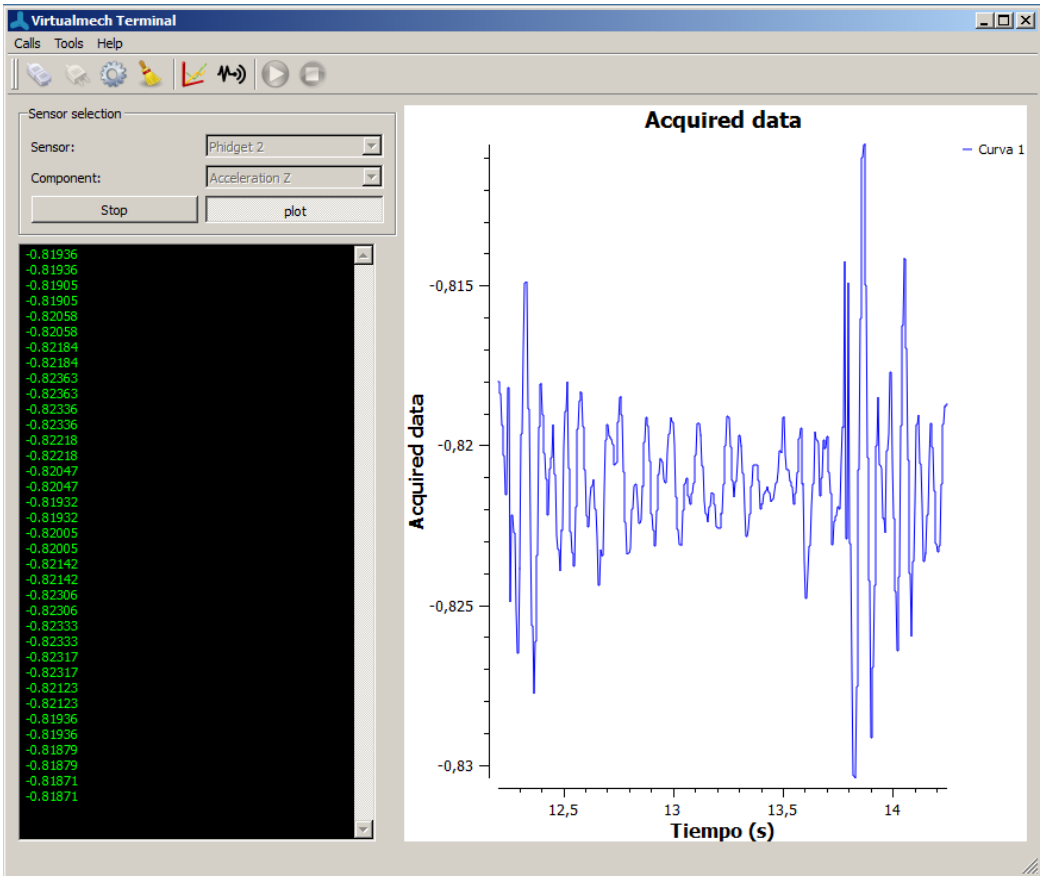

Figura 4.3: Gráfica de la aplicación de adquisición de datos

La sincronización del sistema tacómetro con el resto de sensores se realiza guardando instante de tiempo en el que se inicia la adquisición y el tiempo que tarda el tacómetro en enviar los primeros datos (aunque este tiempo es en la práctica despreciable).

4.2.1. Sistema tacómetro y sincronizador de cámaras de vídeo

Como se ha mencionado anteriormente el sensor magnético no tiene salidas compatibles con los puertos de un PC. Por esta razón, se debe incluir electrónica adicional de diseño y fabricación propias que sea capaz de leer el sensor magnético y controlar el flujo de datos al PC.

El corazón de esta electrónica es el microcontrolador mbed LPC11U24 de NXP. ARM mbed es una plataforma de prototipado rápido quue facilita el montaje y programación de distintos microcontroladores. La placa de prototipo utilizada es la que integra el microcontrolador LPC11U24. Se trata de un microcontrolador de 32 bit y 48 MHz programable por un puerto USB. El microcontrolador se integra en una pequeña placa en formato PDIP con los circuitos de alimentación, conexión con el PC, memoria externa para guardar archivos binarios y otras funciones ya diseñados e integrados. Como se puede ver en la figura 4.4, entre la salidas de la placa se dispone de un puerto serie, un puerto I^2C, un puerto USB, dos puertos SPI, ocho canales de entrada a un conversor ADC y una gran cantidad de puertos GPIO. El microcontrolador puede ser alimentado directamente mediante el USB de programación y comunicación con el PC. Tiene dos pines para alimentar perififéricos de 3,3 V y 5 V.

La plataforma mbed ofrece un compilador online con el que programar todos los microcontroladores compatibles, sin necesidad apenas de hacer cambios para migrar el código de un microcontrolador a otro. El lenguaje utilizado en la programación es C++.

Se ha fabricado una pequeña placa electrónica que integra la plataforma mbed. Esta placa (ver figuras 4.5 y 4.6) contiene los componentes y conectores necesarios para facilitar el montaje en los ensayos.

Además se han añadido dos led en la cara superior de la caja de la placa que son usados como indicadores led. Estos led se encienden cuando se detecta un campo magnético, esto es de gran utilidad a la hora de colocar los sensores magnéticos.

La conexión entre la plataforma mbed y los sensores magnéticos es muy simple. El pin de alimentación del sensor se conecta al pin VOUT (ver figura 4.4) de la plataforma mbed, y uniendo las tierras ya queda alimentado el

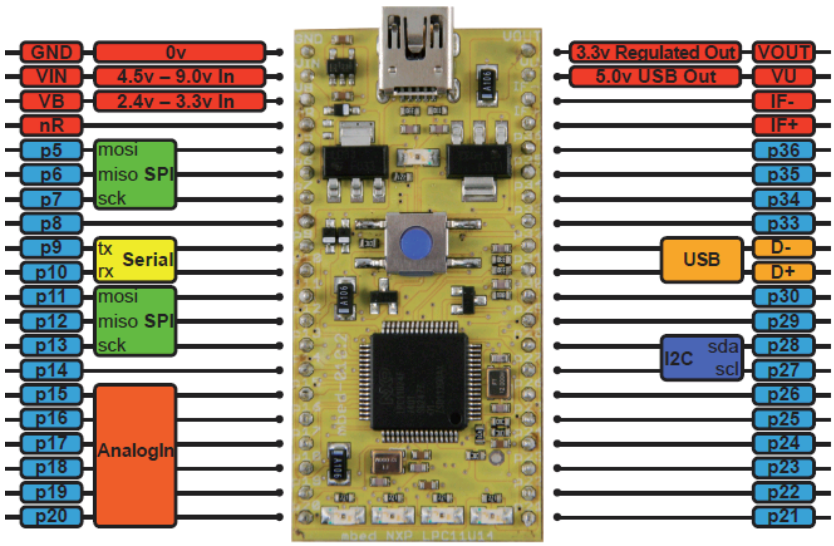

Figura 4.4: Pinout mbed LPC11U24 [14]

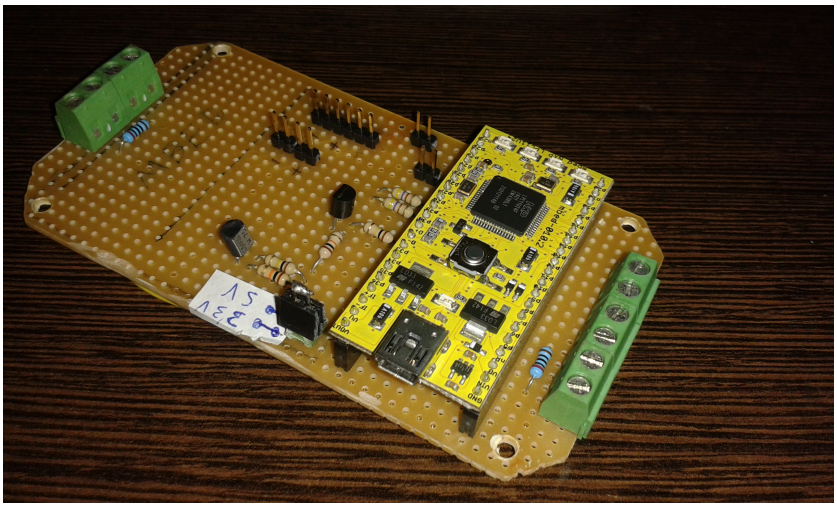

Figura 4.5: Placa electrónica del tacómetro

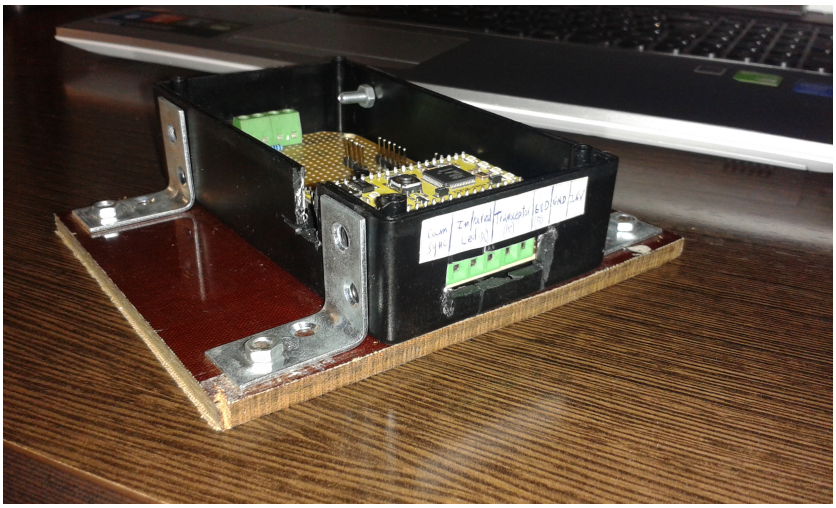

Figura 4.6: Tacómetro. Caja y conectores

sensor. El pin de señal del sensor se puede conectar a cualquier pin GPIO de la plataforma. En este caso se ha conectado un sensor en el pin 19 y el otro en el pin 20.

El programa del microcontrolador que permite leer los sensores magnéticos se describe a continuación:

1. Se configura la velocidad del puerto serie en 460800 baudios, más adelante se explica la elección de esta velocidad.

2. Se inicializan los valores de la señales. La salidas digitales que van conectadas a los leds se ponen a 0 para que estén apagados por defecto. También se inicializa el valor de la señal de sincronización de las cámaras.

3. Una vez inicializado y configurado, el microcontrolador entra en un bucle infinito en el que comprueba constantemente la llegada de bytes procedentes del PC. Cuando se recibe el caracter 'a', que indica inicio de adquisición, el programa activa los *ticker* de lectura de sensores y el *ticker* de control de la señal de sincronización de las cámaras. Si se recibe una 's', que indica fin de la adquisición, se eliminan los *ticker* y se resetean los valores de las señales como al inicio del programa por si se volviese a recibir una 'a', continuando la adquisición. De esta manera, el microcontrolador es controlado por el PC y el PC puede sincronizar todos los sensores. El PC sólo tiene que enviar un caracter 'a' al tacómetro para comenzar a recibir datos de este y guardar el instante de tiempo en el que envía el caracter. Aunque existe un tiempo de transmisión del caracter, es en la práctica despreciable frente a los periodos de adquisición de los sensores.

Los *ticker* son la herramienta que ofrece la plataforma mbed para el control de interrupciones periódicas. Al crear un *ticker*, este se encargará de interrumpir el programa principal cada periodo de tiempo especificado y ejecutar la función asociada al *ticker*.

```
leeSensor.attach_us(&lecturaSensores, TREAD); // TREAD periodo
    en microsegundos
sincro.attach_us(&sincroniza, TSYNC); // TSYNC periodo en
    microsegundos
```

La función *lecturaSensores* es la encargada de la lectura de sensores. esta función envía cuatro bytes por el puerto serie:

1. Se envía un byte con el valor '1', que indica que el próximo byte es el valor del sensor magnético 1.

2. Se lee la entrada digital del sensor 1 y se envía el valor por el puerto serie. También se actualiza el valor de la salida conectada al led indicador del sensor 1.

3. Se envía un byte con el valor '2', que indica que el próximo byte es el valor del sensor magnético 2.

4. Se lee la entrada digital del sensor 2 y se envía el valor por el puerto serie. También se actualiza el valor de la salida conectada al led indicador del sensor 2.

El código que realiza esta función es el siguiente:

```
void lecturaSensores() {
        pc.putc('1');
        pc.putc(sensor1);
        led1 = sensor2;

        pc.putc('2');
        pc.putc(sensor2);
        led2 = sensor1;
}
```

La razón por la que se configura el puerto serie a 460800 baudios está directamente relaccionada con la cantidad de bytes enviado por esta función y la frecuencia a la que es ejecutada. Para justficar esta frecuencia debe calcularse con qué frecuencia máxima pueden pasar los imanes y escoger una frecuencia de lectura, como mínimo, diez veces mayor.

En las ruedas del vehículo se van a colocar 16 imanes equiespaciados. La velocidad del vehiculo máxima es de 19,44 m/s (70 km/h) y el radio de

la rueda mide aproximadamente 0,3 m. Dividiendo la velocidad máxima del vehículo entre la longitud de la circunferencia exterior de la rueda obtenemos una medida aproximada de las revoluciones por segundo de las ruedas a la máxima velocidad, en este caso:

$$\frac{19,44m/s}{2\pi 0{,}3m} = 10rev/s \tag{4.1}$$

Al tener 16 imanes la rueda, el sensor magnético detéctará pasar 160 imanes por segundo a la velocidad máxima, por tanto, la mínima frecuencia de adquisición, es decir, la mínima frecuencia de llamada a la función de la lectura de los sensores debe ser de 1600 Hz. Como también es importante el arco de circunferencia detectado por el sensor al pasar un imán, se ha elegido una frecuencia de un orden mayor para evitar que no se detecte un imán, 10000 Hz.

La función de lectura de sensores envía cuatro bytes por el puerto serie, y es llamada 10000 veces por segundo, por lo que son envíados 40000 bytes por segundo. Si aplicamos la ecuación 3.3 se calcula una velocidad mínima del puerto serie de 400000 baudios. Por esta razón el puerto ha sido configurado a 460800 baudios.

Crear una señal cuadrada periódica con un microcontrolador mbed es tan simple como crear un *ticker* que invierta el estado de una señal digital en una de sus salidas de sus pines GPIO:

```
void sincroniza() {
outS = !outS;
outS2 = !outS2;
}
```

La frecuencia con la que se llama a esta función debe ser el doble de la frecuencia de adquisición de las cámaras porque la cámara sólo comenzará la exposición a la luz en los flancos de subida o bajada de la señal, no en ambos flancos. Como existe una gran cantidad de pines GPIO se ha decidido crear una señal independiente de sincronización para cada cámara. El tiempo que tarda el microcontrolador en invertir una señal y pasar a invertir la de la otra cámara es despreciable en las escalas de tiempo de este proyecto.

Los pines GPIO del microcontrolador configurados como salida tienen una tensión de 3v3 V cuando están en nivel alto. Sin embargo, la señal cuadrada de sincronización de las cámaras debe ser de 24 V. Para solucionar este problema se ha montado el circuito de la figura 4.7 para cada señal de sincronización usando una fuente de alimentación de 24 V.

Los valores de las resistencias del circuito de la figura 4.7 han sido elegidos para conseguir que el transistor funcione en saturación. De esta forma, cuando

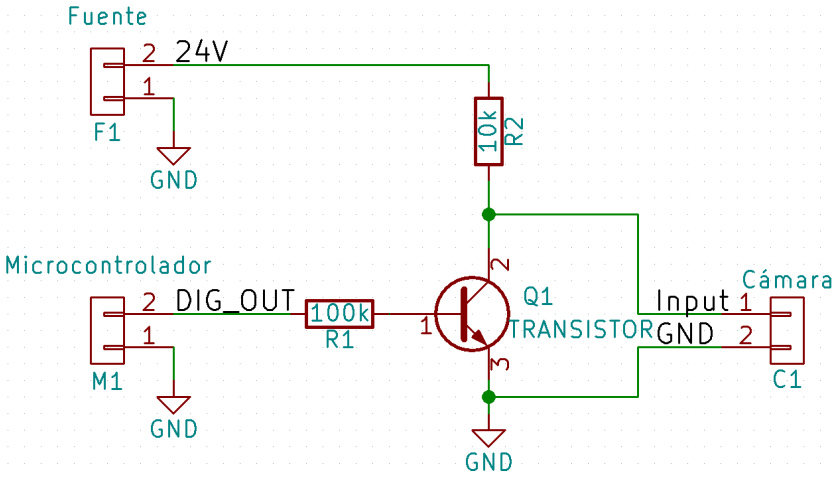

Figura 4.7: Circuito adaptador de tensión de las señales de sincronización de las cámaras

la salida del microcontrolador cambia a nivel alto, el transistor deja pasar corriente entre el colector y el emisor, existiendo una tensión entre ambos puntos de 0 V. Cuando el microcontrolador pone a nivel bajo la salida, el transistor se abre, con una tensión de 24 V entre el colector y el emisor, es decir, a la entrada de la señal de sincronización de la cámara. Esto quiere decir que el circuito invierte la señal de salida del micro, pero la eleva a la tensión requerida por la entrada de la cámara. Para que la señal no llegue invertida a la entrada de la cámara y comience desde los 0 V, sin desfase, se han inicializado las señales de sincronización a nivel alto, que al ser invertidas por el circuito, llegarán 0 V a las cámaras.

El código completo del sistema electrónico del tacómetro puede encontrarse en el apéndice A, totalmente comentado.

4.3. SAD con etapa intermedia

La segunda propuesta para la adquisición de datos se basa en introducir un sistema electrónico con microcontroladores que lea, sincronice y regule el flujo de datos provenientes de los sensores, quedando el PC con la única tarea de recibir los datos y almacenarlos. Esta forma de diseñar el SAD del proyecto tiene numerosas ventajas:

1. La principal ventaja es el control sobre el proceso de adquisición que se consigue mediante electrónica especializada para realizar esta tarea. La primera propuesta utiliza un PC con un sistema operativo (windows)

que es el que controla todo el hardware y en numerables ocasiones el acceso a un nivel bajo de control de hardware es complejo o no está permitido. Es por ello que en los ensayos realizados se producen picos en los periodos de adquisición. Además, a pesar de la gran potencia de procesado de un PC comparado con un microcontrolador común es muy alta, este tiene que atender muchas otras tareas, como el control de periféricos o los recursos que consume el propio sistema operativo. Existen dos formas de solucionar este problema, una es usar un sistema operativo de tiempo real que asegure los tiempos de lectura y escritura con una precisión aceptable y otra es diseñar una electrónica especializada como se ha propuesto en este proyecto. Los microcontroladores permiten el acceso a todos sus procesos a nivel hardware y un control muy preciso del tiempo que va tardar en realizarse cada tarea y el orden en el que se relizan.

2. El PC queda libre de la tarea de adquisición y puede dedicar sus recursos al procesado de los datos.

3. Sincronización. La sincronización con un microcontrolador es muy sencilla, sólo se tiene que acompañar cada dato con una marca de tiempo, leyendo el reloj del microncontrolador en el instante en el que se recibe el dato.

4. Comunicación directa con los sensores. En la propuesta los sensores son leídos utilzando las librerías de funciones que proporciona la API de cada sensor y los driver instalados. Aunque estas funciones suelen estar muy optimizadas, muchas veces no están completamente documentadas, por lo que no es posible conocer como se ejecuta cada función y cuanto tiempo tarda en hacerlo. La abstracción que ofrecen las funciones puede ser una desventaja en proyectos de gran precisión temporal y sincronización como el presente proyecto. La comunicación con el sensor utilizando microcontroladores obliga al aprendizaje de los comandos del sensor y sus protocolos de comunicación, pero a su vez asegura el control de todas las posibilidades del sensor y su sincronización con otros sensores.

Estas ventajas van a repercutir positivamente en la precisión, en la sincronización de los sensores, gracias a la precisión temporal y dedicación de los microcontroladores en la tarea de la adquisición, en la capacidad de control del flujo de datos, pudiendo leer datos de los sensores a una mayor velocidad.

También existen algunas desventajas de este SAD. Utilizar un sistema electrónico especializado aumenta el coste del proyecto. Al tener más componentes electrónicos también existen mas posibles puntos donde puede fallar

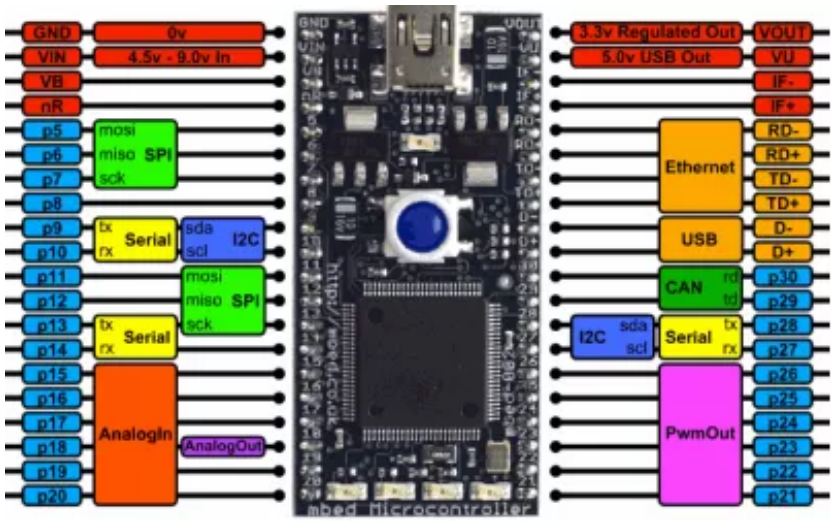

Figura 4.8: Pinout mbed LPC1768 [13]

el sistema, además, el diseño de este sistema requiere un conocimiento más profundo de los sensores y su funcionamiento por lo que el tiempo de desarrollo aumenta (aunque el desarrollo de una aplicación de adquisición como la expuesta en la primera propuesta también es complejo).

Una desventaja adicional es que el sistema propuesto no contempla la adqusición de los sensores de la dinámica vehicular, es decir, de los sensores inerciales de los vagones no instrumentados. Los sensores leídos y sincronizados por este sistema son los tres sensores inerciales, los dos sensores de distancia, el sistema tacómetro y las cámaras.

El microcontrolador utilizado en el SAD es el mbed LPC1768 de NXP. Este microcontrolador pertence a la plataforma mbed. Se trata de una versión con mejores características que el microcontrolador utilizado en el sistema tacómetro (ver sección 4.2.1), también viene montado en una plataforma de desarrollo en formato PDIP y se puede programar con la API online de la plataforma mbed. Este microcontrolador de 32 bit con una frecuencia de 96 MHz tiene tres puertos serie, dos puertos I^2C, un puerto USB, tres puertos SPI, ocho canales de entrada a un conversor ADC, ocho salidas PWM (*Pulse Width Modulation*), una salida analógica y una gran cantidad de pines GPIO. El pinout de la plataforma puede encontarse en la figura 4.8.

4.3.1. Arquitectura del sistema

La arquitectura del sistema con etapa intermedia es muy similar a la propuesta en la adquisición integrada en el PC. Como puede observarse en

la figura 4.9 las conexiones de datos de las IMU de auscultación y de los sensores de distancia pasan por un sistema microcontrolador antes de llegar al PC, las IMU *PhidgetSpatial* no están conectadas. (cambios de alimentación de IMU de auscultación y de sensores de distancia). También ha cambiado la vía por la que se alimentan los sensores inerciales de auscultación. En la primera proppuesta se alimentaban directamente a 5 V por el puerto USB, en este caso se alimentan con 24 V de una de las fuentes de alimentación.

El sistema tacómetro funciona exáctamente igual que en la primera propuesta. Aunque esta funcionalidad se podría haber integrado en el sistema microcontrolador, se ha decidido mantener el formato anterior por dos razones:

1. Se aprovecha el trabajo ya realizado y testeado.

2. Es conveniente que el sistema tacómetro no este muy alejado de las cámaras y los sensores magnéticos, ya que los cables no pueden ser muy largos por la caída de tensión. De esta forma la colocación del sistema microcontrolado es independiente de la colocación de las cámaras y el tacómetro y sólo hay que preocuparse de su conexión con el sistema tacómetro.

El sistema electrónico diseñado que controla el flujo de datos será llamado en adelante sistema microcontrolado como se observa en la figura 4.9. Este sistema está compuesto por dos plataformas mbed con el microcontrolador LPC1768.

Cada microcontrolador mbed LPC1768 tiene tres puertos serie, insuficientes para leer los tres sensores inerciales y los sensores de distancia, es por esto que se han utilizado dos microcontroladores, que suman en total seis puertos serie. Por razones de simetría y de distinto tamaño de flujo de datos (los sensores inerciales tienen un flujo de datos mucho mayor que los sensores de distancia) se han distribuido los sensores de la siguiente forma:

1. **mbed 1**: Esta mbed controlará el flujo de datos de el sensor inercial y el sensor de distancia de uno de los lados del vehículo. También controlará el sensor inercial central situado en el *bogie*.

   El sensor de distancia va conectado al puerto serie de los pines 27 y 28 y el sensor inercial en el puerto serie de los pines 9 y 10. El sensor inercial colocado en la parte central del *bogie* es conectado al puerto serie de los pines 13 y 14.

2. **mbed 2**: Esta mbed controlará el flujo de datos de el sensor inercial y el sensor de distancia del otro lado del vehículo. El sensor de distancia va conectado al puerto serie de los pines 27 y 28 y el sensor inercial en el puerto serie de los pines 9 y 10.

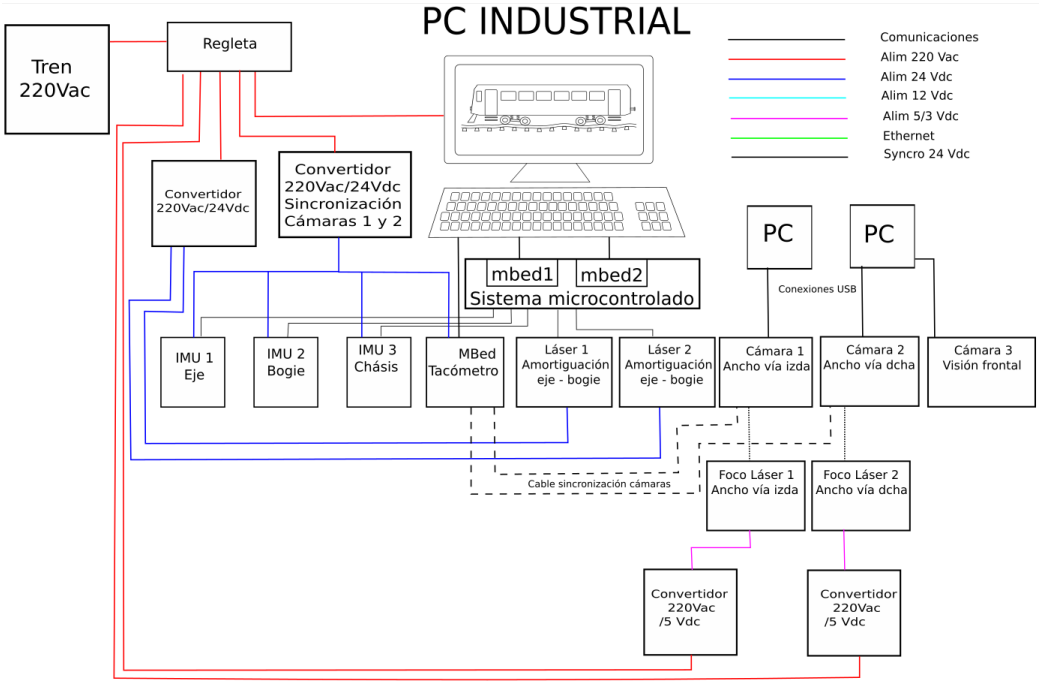

Figura 4.9: Arquitectura de conexiones y alimentación SAD con etapa intermedia

Cada mbed enviará los datos de los sensores conectados a él hacia el PC, es por esto por lo que existen dos conexiones USB entre el sistema microcontrolado y el PC. Por estas mismas conexiones USB se alimentan las plataformas mbed. Se puede ver el aspecto de la placa con los microcontroladores y los conectores para los sensores en la figura 4.10.

Un problema que puede aparecer por el uso de dos microcontroladores distintos, cada uno con su propio temporizador, es una mala sincronización entre los sensores leídos por un microcontrolador y el otro. A priori, este problema no debería aparecer ya que cuando el PC manda los comandos de inicio de adquisición a los microcontroladores, el tiempo entre que llega el mensaje al primer microcontrolador y luego al segundo, es de decenas de microsegundos, un tiempo despreciable en la escala de tiempo del proyecto. Sin embargo, en una futura mejora podría solucionarse de forma elegante este problema mediante una conexión serie entre ambos microcontroladores por alguno de los puertos SPI disponibles, de forma que nada más comenzar establezcan una pequeña comunicación en la que uno de ellos (el esclavo) sincronice su reloj al del otro (el maestro), esta sincronización tiene una precisión del microsegundo, con la programación correcta y la aplicación de los conocimientos de los tiempos de retardo de los puertos SPI de los

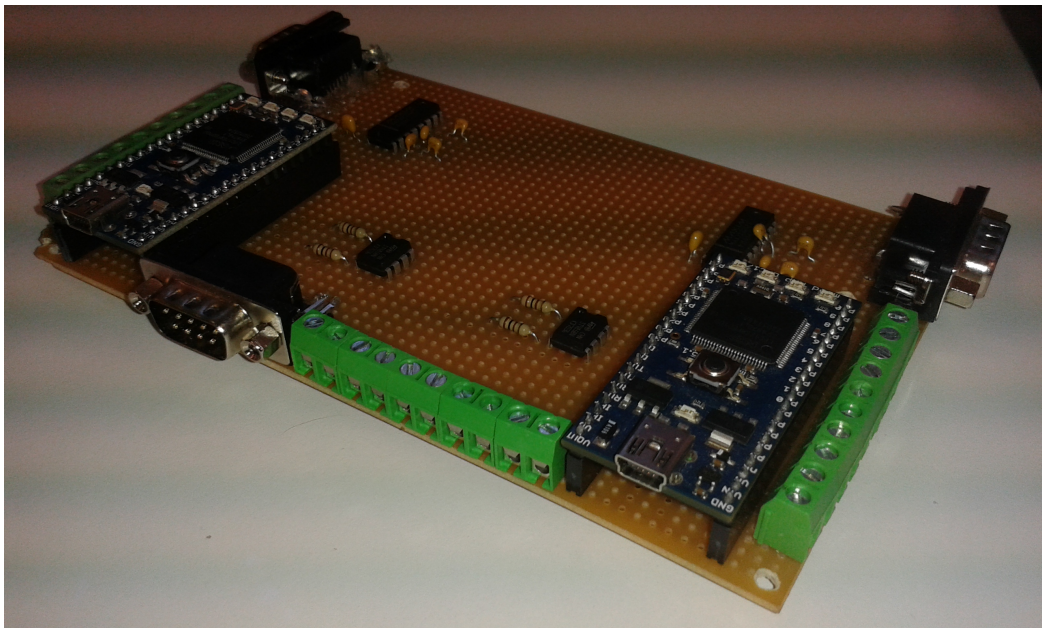

Figura 4.10: Placa electrónica de adquisicón de datos

microcontroladores.

La conexión entre los sensores inerciales y de distancia, y los microcontroladores, no es inmediata debido a los distintos niveles de tensión utilizados por los sensores y los puertos serie UART de los microcontroladores.

4.3.2. Conexión sensor de distancia - microcontrolador

Los niveles de tensión con los que se comunica el sensor (sigue las especificaciones de RS422) no son compatibles con los niveles TTL de los microcontroladores. Para la conversión de los niveles de tensión se ha utilizado el integrado ST485 (también serviría el MAX485, MAX3485 o similares).

Como puede observarse en la figura 3.11, el puerto serie del sensor tiene cuatro pines (Tx+, Tx-, Rx+ y Rx-), dos para la recepción y dos para la transmisión. Como no se va a enviar ningún dato al sensor solo se utilizarán los de transmisión como podemos ver en el esquema de la figura 4.11.

El integrado va alimentado con la salida de 5V del microcontrolador. Las dos resistencias conectadas a los pines TX y RX sirven para limitar la corriente y proteger dichos pines.

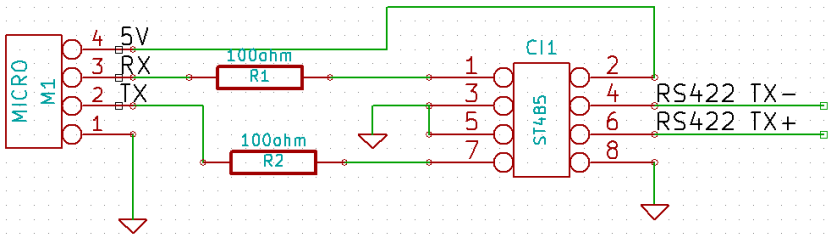

Figura 4.11: Esquema de conexión entre la mbed y el sensor de distancia

4.3.3. Conexión sensor inercial - microcontrolador

El puerto serie de este sensor también tiene diferente niveles de tensión que la UART de la mbed. En este caso el sensor sigue las especificaciones RS232. Para hacer la conversión a niveles TTL se ha usado el integrado MAX3232.

En la figura 3.18 del capítulo anterior se mostraba el pinout del sensor. Se van a utilizar los pines Rx y Tx del puerto serie RS232, además, debe conectarse la tierra y la alimentación. Llama la atención que el sensor tiene dos pines con los que se puede alimentar el sensor. El pin tres es el pin por el que se alimenta cuando se usa la comunicación USB. Sin embargo, para comunicarse por el puerto serie RS232 debe alimentarse por el pin seis con una tensión entre 5,2 V y 36 V. En la figura 4.12 se muestra el esquema de conexiones de estos pines (mostrando, además, su posición en el terminal *DB9* del cable del sensor) con el integrado que hace la conversión de niveles de tensión y de este con el microcontrolador.

4.3.4. Lectura de sensores

La precisión que ofrece un microcontrolador para ponerle una marca de tiempo a un dato recibido de un sensor, junto al objetivo de este SAD de obtener mejores frecuencias de adquisición que la primera propuesta, específicamente la máxima de cada sensor, ha llevado al diseño del SAD para una adquisición por *streaming*. En este método de adquisición el sensor envía datos al sistema microcontrolado a la máxima velocidad posible automáticamente, sin necesidad de que el sistema microcontrolado le envíe comandos solicitando datos, como ocurría con el PC. La frecuencia de adquisicón con este método de lectura es considerablemente mayor: el sensor de distancia mandará datos a 750 Hz (en el sistema integrado se llegaba a los 200 Hz) y el sensor inercial a 1000 Hz (en el sistema integrado se llegaba a los 166 Hz). Además, como el sistema microcontrolado sólo está pendiente de la llegada de datos de los sensores, no tiene que cumplir otras tareas en paralelo, no se

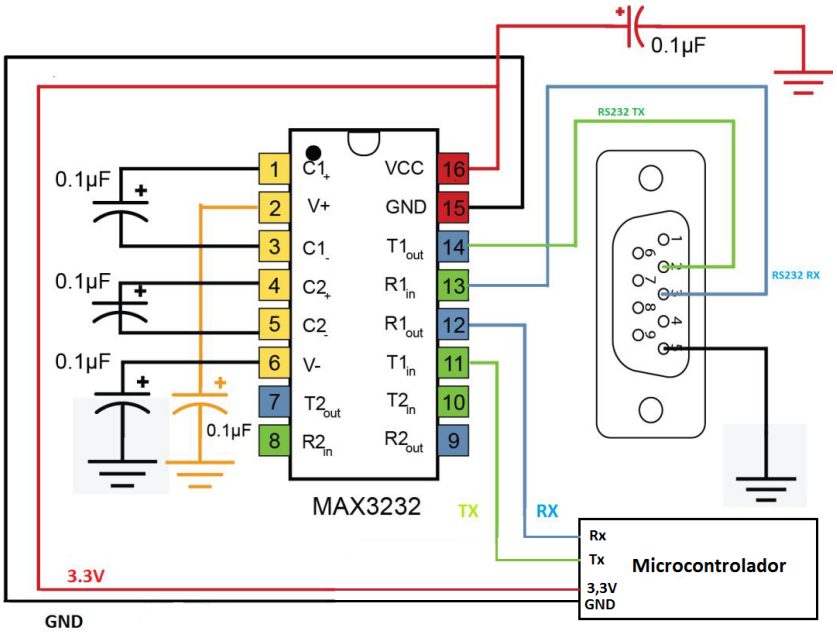

Figura 4.12: Esquema de conexión entre la mbed y el sensor inercial

producen discontinuidades en el periodo de adquisición de los sensores.

La función del microcontrolador queda simplificada. Tiene que recibir los datos, hacer un procesado de estos, y enviarlos junto a su marca de tiempo al PC en formato binario.

El programa de los microcontroladores se ha compilado en la API online de mbed. El código completo puede encontrarse comentado en el apéndice A, totalmente comentado. EL programa comienza con la declaración de las variables y la configuración de la velocidad de los puertos serie:

a. El puerto serie de comunicación con el sensor de distancia se configura a 115200 baudios, que es la velocidad por defecto del sensor.

b. El puerto serie de comunicación con el sensor inercial se configura a 460800 baudios. Como se describe en la sección 3.4, el sensor envía 34 bytes por paquete de datos, a 1000 Hz, son 34000 bytes por segundo, que aplicando la ecuación 3.3 se calcula una velocidad mínima de configuración del puerto serie de 340000 baudios. La velocidad mínima superior 340000 configurable en el puerto serie es de 460800.

c. El puerto de comunicación con el PC debe ser capaz de enviar los datos de todos los sensores. Como luego veremos, el paquete de el sensor inercial que se envía al PC es de 30 bytes (a 1000 Hz) y 8 bytes del paquete con los

datos del láser (a 750 Hz). Esto suma en total $30x1000 + 8x750 = 36000$ bytes por segundo que el microcontrolador envía al PC por el puerto serie. Aplicando la ecuación 3.3 la configuración mínima de velocidad debe es de 360000 baudios, por ello, se ha configurado a 460800.

Una vez hechas las inicializaciones y configuraciones necesarias, el programa entra en un bucle infinito, en el que se realizan las siguientes acciones:

1. Se comprueba si han llegado los bytes procedentes del PC. De forma similar al tacómetro, un byte con el caracter 'a' indica el comienzo de la adquisición y la 's' el final de esta. Cuando se inicia la adquisición se activa una variable semáforo que permite la reconstrucción y envío al PC de los datos de los sensores, se resetea el temporizador, se inicia y se enciende un led indicador. También se envía un paquete de bytes al sensor inercial que le indica que debe comenzar a enviar datos por *streaming*. Cuando finaliza la adquisición, se apaga el led indicador, se para el temporizador y se resetea.

2. Se comprueba la llegada de bytes del sensor inercial y se realizan las operaciones necesarias. Sólo se hace si esto si no se está procesando un paquete del sensor de distancia.

3. Se comprueba la llegada de bytes del sensor de distancia y se realizan las operaciones necesarias. Sólo se hace si esto si no se está procesando un paquete del sensor inercial.

Cuando llega el primer byte de un paquete de un sensor, se deja de comprobar la llegada de bytes de otros sensores. Esto se hace para que no se mezclen los distintos paquetes que se envían al PC con datos de distintos sensores, ya que cuando llega un byte de un sensor, a continuación se procesa y se envían los bytes correspondientes al PC. El tiempo que tarda el microcontrolador en recibir todos los bytes del paquete enviado por un sensor es muy pequeño. Si durante este pequeño periodo de tiempo llegara un byte procedente de otro sensor, este simplemente se guarda en el *buffer* hardware del puerto serie del microcontrolador hasta que se vuelve a preguntar por la llegada de un dato a ese puerto, por lo que no existe pérdida de bytes.

4.3.4.1. Procesado del paquete del sensor de distancia

En la sección 3.3 se describió el formato del paquete del sensor de distancia y como debía ser reconstruido. Una vez leído el byte del sensor de distancia, lo primero que hace el programa es distinguir si es el byte más significativo

o el menos signitivo. Esto es sencillo gracias a que el byte más significativo siempre tiene el bit más significativo a 1 y el byte menos signiticativo siempre lo tiene a 0.

Una vez se ha comprobado que el byte es el más significativo se guarda en una variable entera y se pone a 0 el bit más significativo, para que no influya en el valor reconstruido:

```
distancia = byteIN;
distancia &=~0x80;      // El bit más significativo hay que
    ponerlo a 0
```

La variable *byteIN* guarda el byte recibido. Una vez hecho esto se activa una variable semáforo que permite terminar la reconstrucción con el byte menos significativo. Esto asegura que ya se ha procesado un byte más significativo (el primero que envía el sensor) antes de calcular la distancia.

Cuando se recibe el byte menos significativo, se lee el tiempo. El tiempo que pasa entre la llegada del primer y del segundo byte es despreciable en la escala de tiempo del presente proyecto por lo que puede leerse el tiempo cuando llega el primer o segundo byte del sensor sin pérdida de precisión.

Antes de enviar el paquete con la información recogida hay que terminar la reconstrucción de la medida de la distancia. Para conseguirlo, los siete bit más significativos guardados anteriormente en la variable distancia se desplazan siete posiciones a la izquierda, y al resultado se le suma directamente el valor del byte recibido (es equivalente a hacer una operación OR entre bit de ambas variables en este caso), el byte menos significativo. Con esto obtenemos una reconstrucción como la de la figura 3.14. El código que realiza esta reconstrucción es el siguiente:

```
distancia <<= 7;
distancia += byteIN;      // Unidades de ingeniería
distancia = distancia*0.0498;       // mm
```

El valor reconstruido es un valor en unidades de ingeniería (valor devuelto por el ADC interno del sensor), para su conversión a unidades reales en *mm* debe ser multiplicado por 0,0498 como se vió en la sección 3.3.

Una vez reconstruido el valor de la distancia, se divide en dos bytes, primero se guarda el byte menos significativo, se hace un desplazamiento de los bit ocho posiciones a la derecha, y se guarda el byte más significativo. Todos los bytes que se van a enviar se guardan en un *buffer* dónde se guarda el paquete del sensor de distancia. Los dos primeros bytes de este paquete son dos caracteres 'L'. Esto es así para que el PC sea capaz de diferenciar este tipo de paquetes de los paquetes del sensor inercial. Los últimos cuatro bytes del paquete son los bytes de la variable del tiempo leído anteriormente,

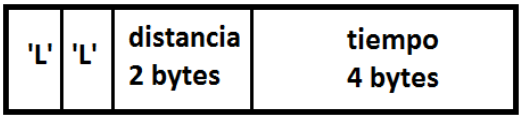

Figura 4.13: Formato del paquete de datos del sensor de distancia enviado al PC

quedando el paquete con los datos del sensor de distancia enviado al PC con el formato de la figura 4.13.

4.3.4.2. Procesado del paquete del sensor inercial

En la sección 3.4 se describió el formato del paquete del sensor inercial y los bytes que contienen los datos de aceleraciones y velocidades angulares.

Debido a la complejidad del paquete enviado por el sensor inercial, se va a utilizar una variable que almecene el número de bytes recibidos por el sensor que pertenecen a un mismo paquete (variable *nBytes*).

Una vez recibido un byte por el puerto serie conectado al sensor inercial, primero debe comprobarse que el byte recibido corresponde a los bytes de inicio de paquete y no se están recibiendo bytes erróneos o de datos. Recordar que los paquetes enviados por estos sensores comenzaban siempre con los bytes 'u', 'e', por lo que cada vez que se recibe un byte se comprueba si es una 'e' y el anterior una 'u' (cada vez que llega un nuevo byte se guarda el anterior). Además, para asegurarse que no son dos bytes de datos de un paquete que coincidan con estos caracteres, se comprueba que el número de bytes recibidos del último paquete es igual o mayor al número de bytes de los paquetes enviados por el sensor (variable *nBytes*). Es por esto por lo que se inicializa la variable *nBytes* con el tamaño del paquete del sensor, sino se hiciera esto nunca comenzaría la reconstrucción de ningun paquete aunque sea recibido correctamente por el puerto serie.

Una vez se reciben caracteres 'u', 'e', se envía por el puerto serie dos caracteres 'I', se guarda el tiempo y se cuentan los bytes recibidos de ese paquete entrante. Cada vez que se recibe un byte de datos de aceleraciones o velocidades angulares (los bytes de datos son los bytes de la lista de la sección 3.4) simplemente se reenvía hacia el PC, el resto de bytes son ignorados.

Cuando se recibe el último byte de datos, se reenvía y además se envía el tiempo guardado anteriormente dividido en 4 bytes, enviando primero el byte más significativo. El paquete resultante enviado al PC tiene el formato mostrado en la figura 4.14.

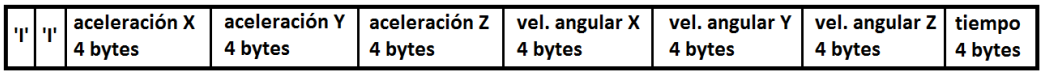

Figura 4.14: Formato del paquete de datos del sensor inercial enviado al PC

Capítulo 5

Resultados

En este capítulo se van a mostrar resultados de distintos sensores para desmostar el funcionamiento correcto del SAD descrito en la sección 4.3

5.1. Adquisición sensor magnético

El sistema tacómetro descrito en la sección 4.2.1 es el encargado de recoger los datos de los sensores magnéticos que detectan el paso de los imanes colocados en las ruedas. Como se describe en dicha sección la frecuencia de paso de imanes es de 160 Hz. Sin embargo, la frecuencia elegida en los ensayos para la lectura del sensor es de 10 kHz. Debido a esto, era esperable encontrar varias lecturas positivas por cada imán detectado (el sistema devuelve 1 cuando se detecta el campo magnético de un imán y 0 cuando no es detectado). Efectivamente, se encuentran grupos de alrededor de veinte valores positivos cada vez que los sensores detectan un imán.

Se ha realizado un programa en matlab que analiza los datos de los sensores magnéticos de un ensayo. Para ello, sabiendo que el tiempo entre cada dato es de $1/10kHz = 100\mu s$, se guarda el instante de tiempo en el que ha sido detectado un imán (esto es, un grupo de unos). Para calcular la posición, se suma una decimosexta parte de la circunferencia total de la rueda (cada rueda tiene 16 imanes) cada vez que se detecta un imán. Para calcular la velocidad se ha dividido la decimosexta parte de la circunferencia total de la rueda entre el tiempo pasado entre un imán y el anterior detectado.

Se pueden observar los resultados de este simple análisis en las figuras 5.1 y 5.2, que muestran la velocidad y posición de ambas ruedas, respectivamente. Los datos de la figura 5.1 han sido previamente filtrados con un filtro paso baja para suavizar el alto ruido.

Debe tenerse en cuenta que las ruedas pueden deslizar, por lo que la

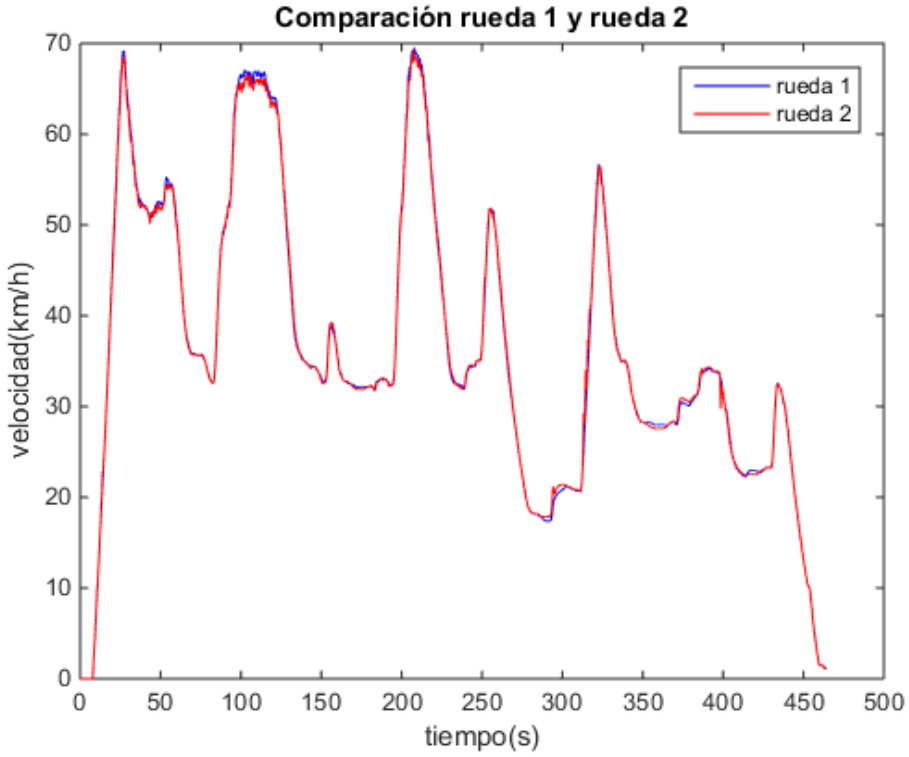

Figura 5.1: Velocidad de cada rueda del vehículo en un ensayo

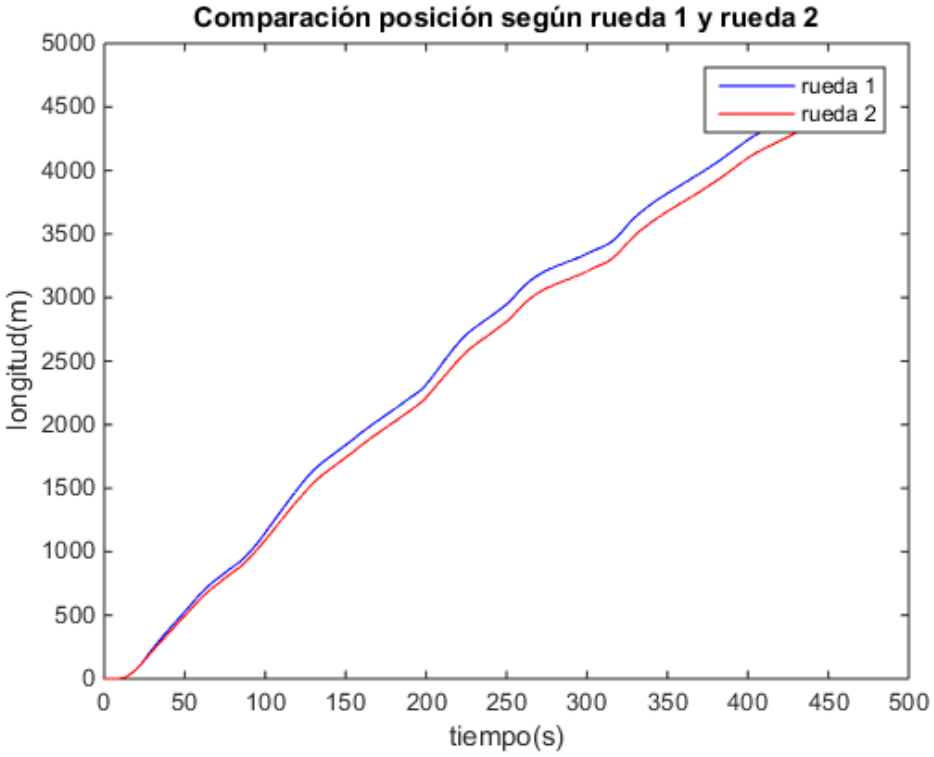

Figura 5.2: Posición de cada rueda del vehículo en un ensayo

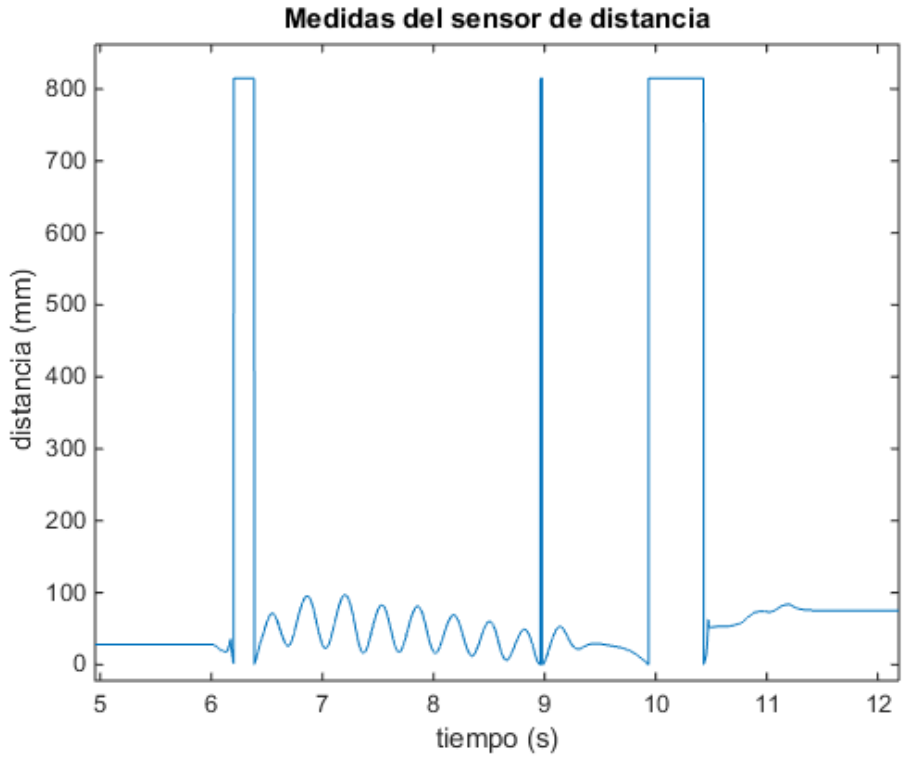

Figura 5.3: Medidas del sensor de distancia

velocidad y posición del vehículo pueden ser algo mayores que la calculada a partir de los datos obtenidos de la velocidad y posición de las ruedas en ciertos puntos.

Sin embargo, estas gráficas si que verifican el buen funcionamiento del sistema tacómetro. Las velocidades máximas obtenidas en los tramos rectos son de 70 Km/h que es exáctamente la máxima velocidad esperada. En las curvas se observan bajadas de velocidad y pequeñas diferencias de velocidad entre ambas ruedas.

5.2. Adquisición sensor de distancia

En la figura 5.3 se muestra una gráfica de una pequeña prueba. En esta prueba simplemente se ha movido un poco el sensor de distancia para ver las oscilaciones, y sacarlo de rango para comprobar los valores devueltos.

Recordar (sección 3.3, figura 3.9) que el sensor devuelve los valores 16372 o 16374 cuando está fuera de rango, que multiplicados por el valor de reconstrucción 0,0498 (sección 3.3) debería enviarse el valor 815 cuando el sensor

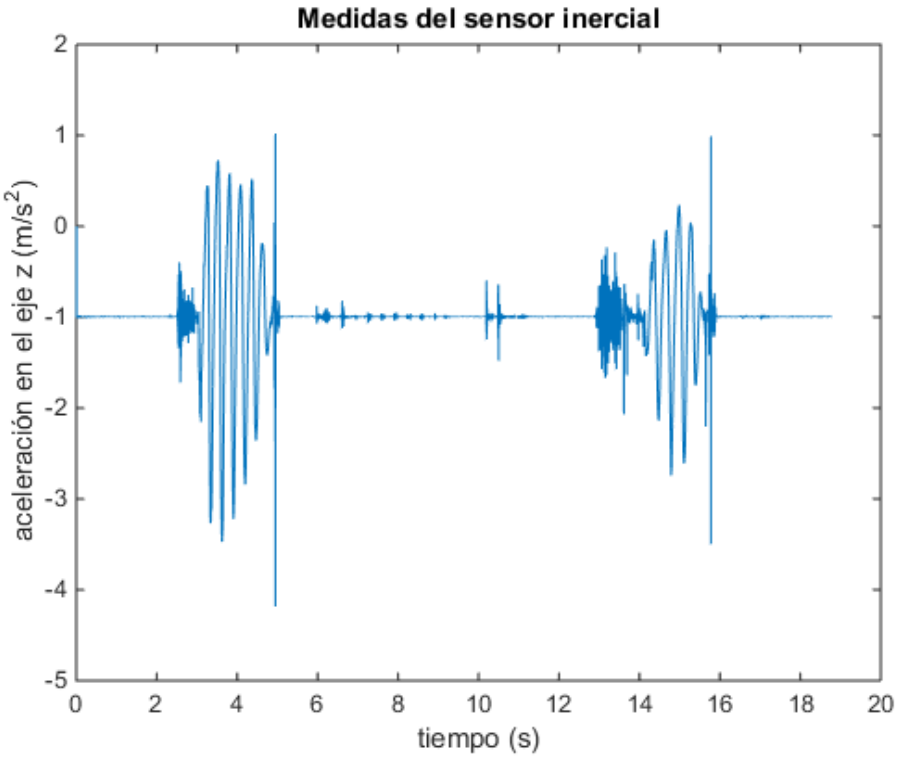

Figura 5.4: Medidas del sensor inercial

está fuera de rango, justo el valor obtenido en la prueba.

5.3. Adquisición sensor inercial

De igual manera, se puede observar en la figura 5.4 una gráfica con las oscilaciones del sensor inercial en la misma prueba que el sensor de distancia. Aunque los sensores han sido leídos en la misma prueba (comprobando así la correcta lectura de ambos sensores simultáneamente), los movimientos de cada sensor han sido independientes.

Apéndice A

Matrices de rotación

Las matrices de rotación 3D respecto a cada uno de los ejes cartesianos son:

$$R_x(\varphi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{pmatrix} \tag{A.1}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \tag{A.2}$$

$$R_z(\psi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.3}$$

Para obtener una matriz de rotación completa podemos multiplicar las matrices A.1, A.2 y A.3 obteniendo una matriz que nos permite realizar cualquier giro que deseemos:

$$R_{xyz}(\varphi,\theta,\psi) = R_x(\varphi)R_y(\theta)R_z(\psi) = \begin{pmatrix} C\psi C\theta & -S\psi C\varphi + C\psi S\theta S\varphi & S\psi S\varphi + C\psi S\theta C\varphi \\ -S\psi C\theta & C\psi C\varphi + S\psi S\theta S\varphi & -C\psi S\varphi + S\psi S\theta C\varphi \\ -S\theta & C\theta S\varphi & C\theta C\varphi \end{pmatrix} \tag{A.4}$$

A.1. Aproximación pequeños ángulos

Cuando los ángulos de giro de la matriz A.4 toman valores pequeños (por debajo de la unidad) esta matriz puede aproximarse por una forma más sencilla. Esto es posible porque el coseño de un ángulo muy pequeño es

aproximadamente 1 y el seno de un ángulo muy pequeño se puede aproximar por el valor del propio ángulo:

$$\begin{aligned} \alpha &<< 1 \\ \cos(\alpha) &\approx 1 \\ \sin(\alpha) &\approx \alpha \end{aligned} \tag{A.5}$$

Aplicando las simplificaciones A.5 en la matriz de rotación A.4 y suponiendo despreciable el producto de dos ángulos respecto a cualquiera de ellos, obtenemos la matriz de rotación simplificada:

$$R_{xyz}(\varphi,\theta,\psi) = \begin{pmatrix} 1 & -\psi+\theta\varphi & \psi\varphi+\theta \\ \psi & 1+\psi\theta\varphi & -\varphi+\psi\theta \\ -\theta & \varphi & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\varphi \\ -\theta & \varphi & 1 \end{pmatrix} \tag{A.6}$$

Apéndice A

Código plataformas mbed

Código completo del archivo *main.cpp* del sistema tacómetro:

```
#include "mbed.h"

// Parametros variables en cada aplicación*****************
#define TSYNC 16666 // Periodo de sincronización en
    microsegundos
#define TREAD 100     // Periodo de lectura de sensores en
    microsegundos

DigitalOut led1(p29);    // Salida digital conectada al led
    indicador 1
DigitalOut led2(p26);    // Salida digital conectada  al led
    indicador 2

Serial pc(USBTX, USBRX);

DigitalIn sensor1(p19);      // Entrada digital del sensor 1
DigitalIn sensor2(p20);      // Entrada digital del sensor 1

Ticker leeSensor;            // Ticker con la función de leer
    periódicamente la salida de los sensores magnéticos
Ticker sincro;               // Ticker con la función de cambiar
    de estado la señal de sincronización

DigitalOut outS(p30);        // Señal de sincronización de las cá
    maras (1)
DigitalOut outS2(p34);        // Señal de sincronización de las c
    ámaras (2)

DigitalOut myled(LED1);      // Led utilizado para debugging

// Función que lee periódicamente la salida de los sensores magn
    éticos
```

```
void lecturaSensores() {
// Se envían los datos de los sensores por el puerto serie al PC
pc.putc('1');          // Se envía un byte con el valor 1 antes del
    valor del primer sensor
pc.putc(sensor1);    // Se envía el valor digital actual del
    sensor 1
//pc.putc(rand()%2);  // Debugging
led1 = sensor2;       // Se actualiza el estado del led indicador
    1 de acuerdo al valor actual del sensor 1

pc.putc('2');          // Se envía un byte con el valor 2 antes del
    valor del segundo sensor
pc.putc(sensor2);    // Se envía el valor digital actual del
    sensor 1
///pc.putc(rand()%2);  // Debugging
led2 = sensor1;       // Se actualiza el estado del led indicador
    2 de acuerdo al valor actual del sensor 2
}

// Función que cambia de estado la señal de sincronización
void sincroniza() {
outS = !outS;    // Se invierte el estado de la salida del pin
outS2 = !outS2;
}

int main() {
pc.baud(460800);     // Configuración de la velocidad de
    comunicación del puerto serie
char car;            // Variable auxiliar en la que se guardará
    el valor del último caracter enviado por el PC
myled = 0;  // Inicialización de señales
led1 = 0;
led2 = 0;
outS = 1;    // La señal que sale del micro se va a invertir al
    pasar por el transistor, por lo que para que la cámara vea un
     flanco de subida al inicio de la captura de imágenes, se
    debe enviar un flanco de subida desde el micro
outS2 = 1;
pc.printf("Listo para enviar.\n\r");
while(1) {  // Bucle infinito
if(pc.readable()){  // Si existe algún byte en el buffer que ha
    enviado el PC
car = pc.getc();     // Se lee el byte
if(car == 'a'){ // Si es el carácter 'a', indica el inicio de la
     adquisición
leeSensor.attach_us(&lecturaSensores, TREAD); // Se activa el
    ticker de lectura de los sensores con periodo TREAD en
    microsegundos
sincro.attach_us(&sincroniza, TSYNC); // Se activa el ticker de
```

```
    control de estado de la señal de sincronización con periodo
    TSYNC en microsegundos
}else if(car == 's'){    // Si es el carácter 'a', indica el
    final de la adquisición
leeSensor.detach(); // Se descativan los ticker
sincro.detach();
myled = 0;  // Se resetean las señales
led1 = 0;
led2 = 0;
outS = 1;    // La señal que sale del micro se va a invertir al
    pasar por el transistor, por lo que para que la cámara vea un
     flanco de subida al inicio de la captura de imágenes, se
    debe enviar un flanco de subida desde el micro
outS2 = 1;
}
}
}
}
```

Código completo del archivo *main.cpp* del sistema microcontrolador:

```
/*************************************
Adquisición de datos IMU y Láser
*************************************/
#include "mbed.h"

#define TAMPAQUETEIMU 34     // Tamaño del paquete recibido de la
     IMU
#define TAMENVIOIMU 30       // Tamaño del paquete enviado al PC
    con los datos de la IMU
#define TAMENVIOLASER 8      // Tamaño del paquete enviado al PC
    con los datos del sensor de distancia

/********************
Funciones
********************/
void startStreamingIMU();    // Manda los bytes a la IMU
    necesarios para que comience a enviar datos por streaming
void envioPaquete(char paquete[], int nElementos); // Envía por
    el puerto serie un paquete de datos
/********************/

DigitalOut led(LED1);    // Led indicador. Se enciende cuando se
    están adquiriendo datos

// Puertos UART a utilizar
Serial pLaser(p9, p10);
Serial pIMU(p28, p27);
Serial pc(USBTX, USBRX);
```

```
// Temporizador para el control del tiempo
Timer t;
unsigned int auxtime = 0;    // Variable auxiliar para la lectura
    del tiempo
unsigned char byteIN, byteINanterior;    // Variables auxiliares
    para guardardar bytes recibidos

// Variables para la reconstrucción de los datos de la IMU
unsigned int nBytes = TAMPAQUETEIMU;
char paqueteEnvio[TAMENVIOIMU]; // 12 bytes para las
    aceleraciones (bytes cada eje) y otros 4 para la marca de
    tiempo y otros 12 para las velocidades angulares más un byte
    'i' que indica que es el paquete de una IMU

// Variables para la reconstrucción datos del láser
unsigned int distancia; // Sólo se usarán los 2 bytes menos
    significativos
char paqueteEnvioL[TAMENVIOLASER];
char fbyte = 0; // Indica si el byte leído es el primero delos
    dos enviados por el láser (tomando valor true) o el segundo (
    tomando valor false)

// Variables semáforo para que no se mezclen los bytes de
    distintos paquetes
bool enviandoIMU = false;
bool enviandoLASER = false;
bool adquiriendo = false;

int main() {
// Inicialización de puertos UART
pc.baud(460800);     // Configuración del puerto conectado al PC
pIMU.baud(460800);  // Configuración del puerto conectado a la
    IMU
pLaser.baud(115200);     // Configuración del puerto conectado al
    sensor de distancia

pc.printf("Listo para recibir datos.\n\r");

while(1) {  // Bucle infinito
        // Se comprueba la llegada de nuevos bytes procedentes
            del PC
        if(pc.readable()){
                char c = pc.getc(); // Se lee el byte
                if(c == 'a'){    // Si es el carácter 'a', indica
                    el inicio de la adquisición
                        adquiriendo = true;
                        t.reset();
                        t.start();
                        startStreamingIMU();
```

```
                led = 1;
        }else if(c == 's'){ // Si es el carácter 'a',
            indica el final de la adquisición
                adquiriendo = false;
                led = 0;
                t.stop();
                t.reset();
        }
}


if(pIMU.readable() && enviandoLASER == false){
        byteINanterior = byteIN;
        byteIN = pIMU.getc();
        if(adquiriendo){    // Sólo se reconstruye y env
            ían datos cuando se ha iniciado la adquisició
            n
        // Reconstrucción del dato
        if(byteIN == 0x65 && byteINanterior == 0x75 &&
            nBytes >= (TAMPAQUETEIMU-1)){    // Si el
            byte recibido es un 75 en hex, este byte
            marca la llegada de una nueva medida
                auxtime = t.read_us();
                // Se pone el contador de bytes de un
                    paquete a 0
                nBytes = 1;
                enviandoIMU = true;      // Se está
                    enviando un paquete de IMU
                pc.putc('I');
                pc.putc('I');
        }else{  // Todo lo que no sea 0x75 es parte del
            resto del dato
                // Se suma un byte mas
                nBytes++;
                // Para reconstruir el dato se mandan
                    primero los bytes más significativos
                    (que también es el orden en el que
                    llegan)
                switch(nBytes){
                case 6:
                pc.putc(byteIN);
                break;
                case 7:
                pc.putc(byteIN);
                break;
                case 8:
                pc.putc(byteIN);
                break;
                case 9:
```

```
                pc.putc(byteIN);
                break;
                case 10:
                pc.putc(byteIN);
                break;
                case 11:
                pc.putc(byteIN);
                break;
                case 12:
                pc.putc(byteIN);
                break;
                case 13:
                pc.putc(byteIN);
                break;
                case 14:
                pc.putc(byteIN);
                break;
                case 15:
                pc.putc(byteIN);
                break;
                case 16:
                pc.putc(byteIN);
                break;
                case 17:
                pc.putc(byteIN);
                break;
                case 20:
                pc.putc(byteIN);
                break;
                case 21:
                pc.putc(byteIN);
                break;
                case 22:
                pc.putc(byteIN);
                break;
                case 23:
                pc.putc(byteIN);
                break;
                case 24:
                pc.putc(byteIN);
                break;
                case 25:
                pc.putc(byteIN);
                break;
                case 26:
                pc.putc(byteIN);
                break;
                case 27:
                pc.putc(byteIN);
```

```
                    break;
                    case 28:
                    pc.putc(byteIN);
                    break;
                    case 29:
                    pc.putc(byteIN);
                    break;
                    case 30:
                    pc.putc(byteIN);
                    break;
                    case 31:
                    pc.putc(byteIN);
                    paqueteEnvio[3] = auxtime;
                    auxtime >>= 8;
                    paqueteEnvio[2] = auxtime;
                    auxtime >>= 8;
                    paqueteEnvio[1] = auxtime;
                    auxtime >>= 8;
                    paqueteEnvio[0] = auxtime;
                    pc.putc(paqueteEnvio[0]);
                    pc.putc(paqueteEnvio[1]);
                    pc.putc(paqueteEnvio[2]);
                    pc.putc(paqueteEnvio[3]);
                    enviandoIMU = false;          // Se ha
                        terminado de enviar el paquete de la
                        IMU
                    break;
                    }
                    }
            }
}

if(pLaser.readable() && enviandoIMU == false){
            byteIN = pLaser.getc();
            if(adquiriendo){     // Sólo se reconstruye y env
                ían datos cuando se ha iniciado la adquisició
                n
            // Si el byte recibido tiene el bit más
                significativo a 1 es el byte más
                significativo
            if(byteIN > 127){
                    fbyte = 1;
                    distancia = byteIN;
                    distancia &=~0x80;     // El bit mas
                        significativo hay que ponerlo a 0
                    enviandoLASER = true;    // Se comienza a
                         enviar el paquete del láser
            }else{  // Recibido byte menos significativo
```

```
                    if (fbyte == 1){    // Si se recibió un
                        byte más significativo puede
                        reconstruirse el valor medido por el
                        laser
                    auxtime = t.read_us(); // Se lee el
                        tiempo cuando en el instante de
                        llegada del primer byte del paquete
                    fbyte = 0;
                    distancia <<= 7;
                    distancia += byteIN;    // Unidades de
                        ingenieria
                    distancia = distancia*0.0498;      // mm
                    // Los dos primeros bytes byte de los
                        paquetes del laser seran una 'L'
                    paqueteEnvioL[0] = 'L';
                    paqueteEnvioL[1] = 'L';
                    paqueteEnvioL[3] = distancia;
                    distancia >>= 8;
                    paqueteEnvioL[2] = distancia;
                    // Se añaden también los 4 bytes del
                        tiempo y se envía el paquete (siempre
                         el byte más significativo se envía
                        primero)
                    paqueteEnvioL[7] = auxtime;
                    auxtime >>= 8;
                    paqueteEnvioL[6] = auxtime;
                    auxtime >>= 8;
                    paqueteEnvioL[5] = auxtime;
                    auxtime >>= 8;
                    paqueteEnvioL[4] = auxtime;
                    envioPaquete(paqueteEnvioL,
                        TAMENVIOLASER);
                    enviandoLASER = false;    // Se ha
                        terminado de enviar el paquete del lá
                        ser
                    }
                }
                }
        }
}
}

void envioPaquete(char paquete[], int nElementos){ // Envía por
    el puerto serie un paquete de datos
        for(int i = 0; i < nElementos; i++){
                pc.putc(paquete[i]);
        }
}
```

```
void startStreamingIMU(){
        // Se envia un paquete de datos a la IMU que indica el
            comienzo de lectura de medidas
        pIMU.putc(0x75); // K
        pIMU.putc(0x65); // A
        pIMU.putc(0x0C); // alt + 12
        pIMU.putc(0x05); // alt + 5
        pIMU.putc(0x05); // alt + 5
        pIMU.putc(0x11); // alt + 11
        pIMU.putc(0x01); // alt + 1
        pIMU.putc(0x01); // alt + 1
        pIMU.putc(0x01); // alt + 1
        pIMU.putc(0x04); // alt + 4
        pIMU.putc(0x1A); // alt + 26
        pc.printf("Paquete de inicio enviado.\n\r");
}
```

Bibliografía

[1] Depaoli, Roberto, Díaz, Daniel, Fernández, Luis y Stockli, Roberto, *El tratamiento de la distorsión radial en metrología efectuada con cámaras digitales*, Departamento de Ingeniería, Universidad Nacional de La Matanza.

[2] Palomo Pinto, Francisco Rogelio, Pérez Vega Leal, Alfredo y Galván Diez, Eduardo, *Problemas resueltos de instrumentación electrónica*, Universidad de Sevilla. Secretariado de publicaciones, 2007.

[3] Platero, Carlos, *Apuntes de Visión Artificial*, Universidad Politécnica de Madrid, Dpto. Electrónica, Automática e Informática Industrial.

[4] MERCÈ BRUNED, *Tecnología de un escáner 3D con visión artificial*, http://www.measurecontrol.com/tecnologia-de-un-escaner-3d-con-vision-artificial/, 2009.

[5] Escalona Franco, José Luis, *Cinemática ferroviaria para su aplicación a sistemas de visión artificial*, comunicación personal, 2015.

[6] Escalona Franco, José Luis, *Desarrollo de nuevas tecnologías de auscultación ferroviaria basadas en la simulación en tiempo real*, comunicación personal, 2015.

[7] Li, M.X.D, Berggren, E.G, Berg, M. y Persson, I., *Assessing track geometry quality based on wavelength spectra and track-vehicle dynamic interaction*, Vehicle System Dynamics, 46, Suppl., 261 -276, 2008.

[8] XIMEA, *USB 3.0 camera series, Technical Manual*, proporcionado por el fabricante, XIMEA, 2015.

[9] Ruiz Arahal, Manuel, *Estimación de la posición de objetos*, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, apuntes Sistemas de Percepción.

[10] MICRO-EPSILON, *optoNCDT 1302, Instruction Manual*, proporcionado por el fabricante, MICRO-EPSILON, 2015.

[11] Lord MicroStrain, *3DM-GX4-25, Data Communications Protocol*, proporcionado por el fabricante, Lord MicroStrain, 2015.

[12] Lord MicroStrain, *3DM-GX4-25, User Manual*, proporcionado por el fabricante, Lord MicroStrain, 2015.

[13] ARM mbed, *mbed LPC1768*, https://developer.mbed.org/platforms/mbed-LPC1768/.

[14] ARM mbed, *mbed LPC11U24*, https://developer.mbed.org/platforms/mbed-LPC11U24/.

[15] García Vallejo, D., *Programa de adquisición de datos de sensores ópticos e inerciales. Manual de uso*, Virtualmech, 2015.